

Trie: дерево, но не такое



- 2001 — 2013: Java @ МФТИ
- 2004 — 2010: Java @ NetCracker
- 2011 — 2011: Java @ Яндекс
- 2011 — 2014: Java @ Одноклассники
- 2014 — 2015: Java @ Republer
- с 2015: Java @ СберТех

Trie

- Что такое Trie
- Как "посчитать" авторов Википедии с помощью Trie
- Как проверить соответствие строки 287 тыс. шаблонов за 1 миллисекунду

Структуры данных в Java

- Можно реализовать любые

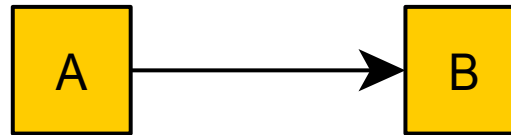
Структуры данных в Java

- Можно реализовать любые
- Уже есть в JDK:
 - *List, Queue, Deque, RandomAccess, SortedList*
 - *Set, SortedSet, NavigableSet*
 - *Map, SortedMap, NavigableMap*

Структуры данных в Java

- Можно реализовать любые
- Уже есть в JDK:
 - ArrayList, LinkedList, ...
 - HashSet, LinkedHashSet, TreeSet, ConcurrentSkipListSet, CopyOnWriteArraySet, ...
 - HashMap, TreeMap, LinkedHashMap, ConcurrentHashMap, ConcurrentSkipListMap, ...

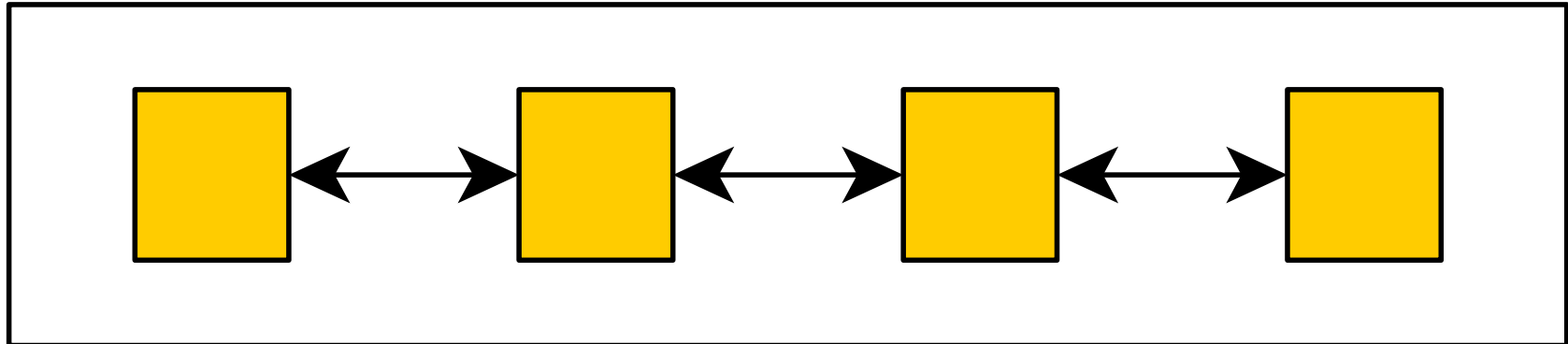
Структуры данных в Java



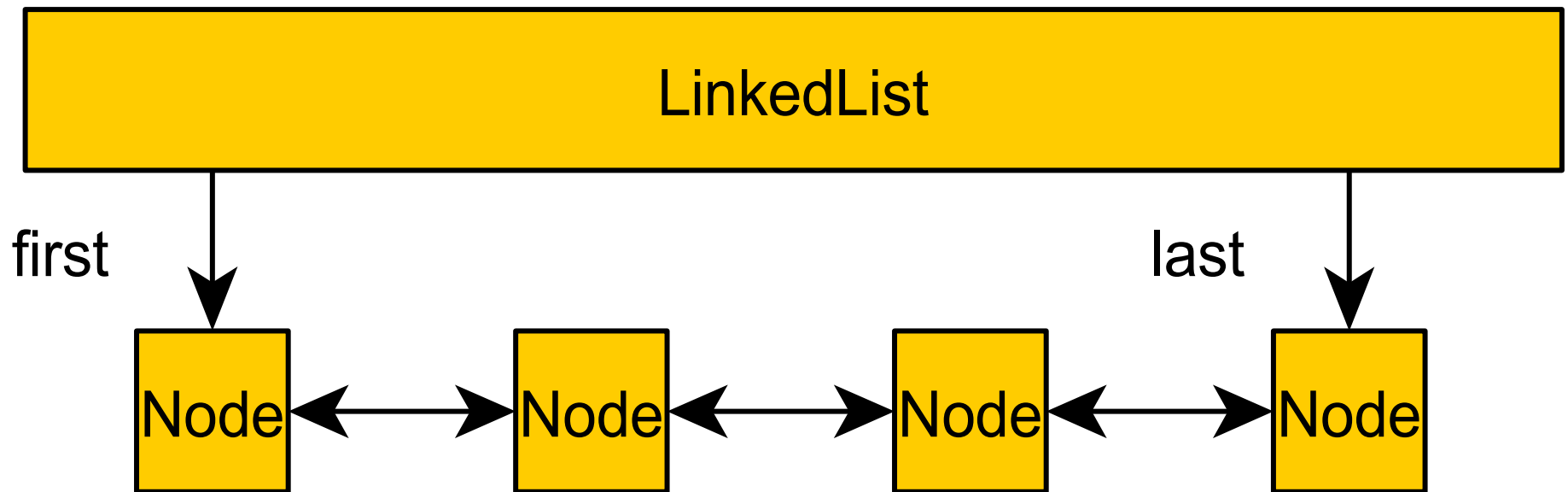
```
public class A {  
    private B b;  
    ...  
}
```

```
public class B {  
    ...  
}
```

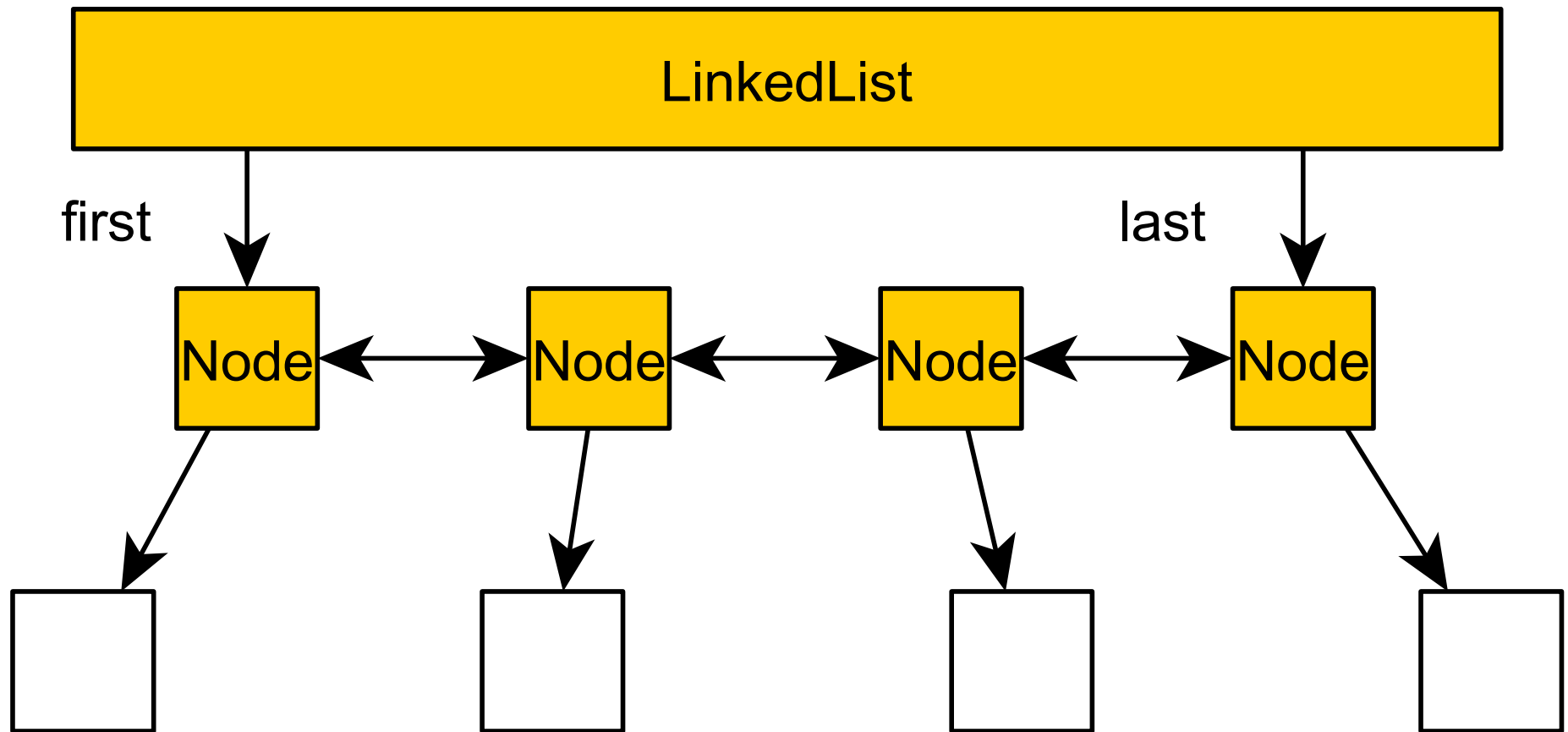
Структуры данных в Java



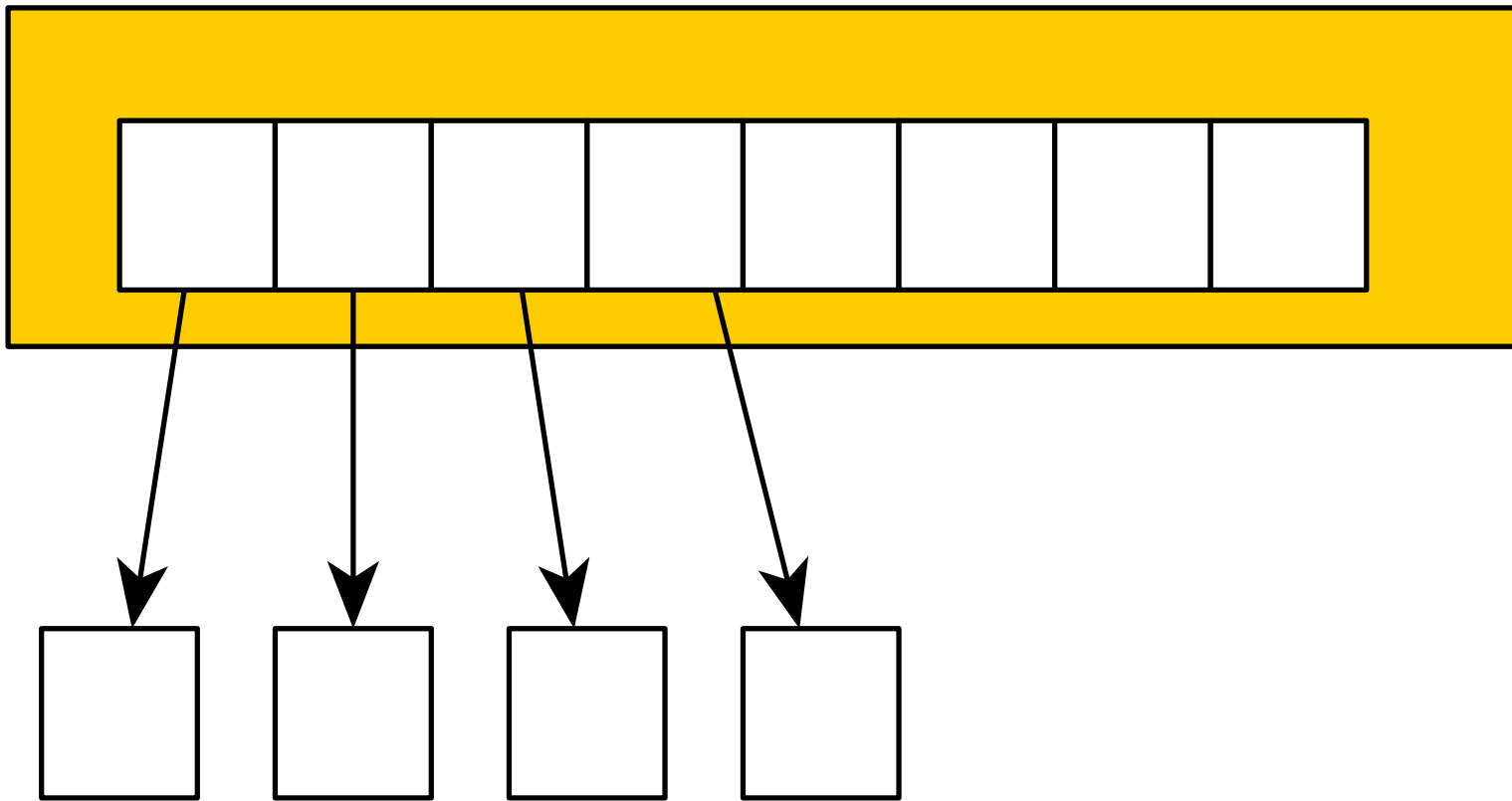
Структуры данных в Java



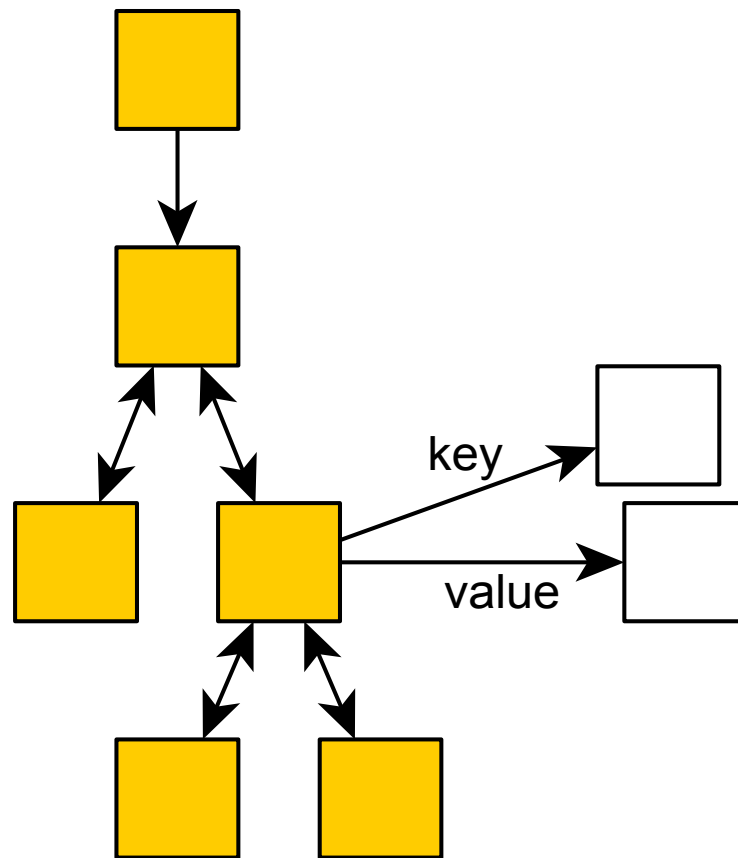
Структуры данных в Java



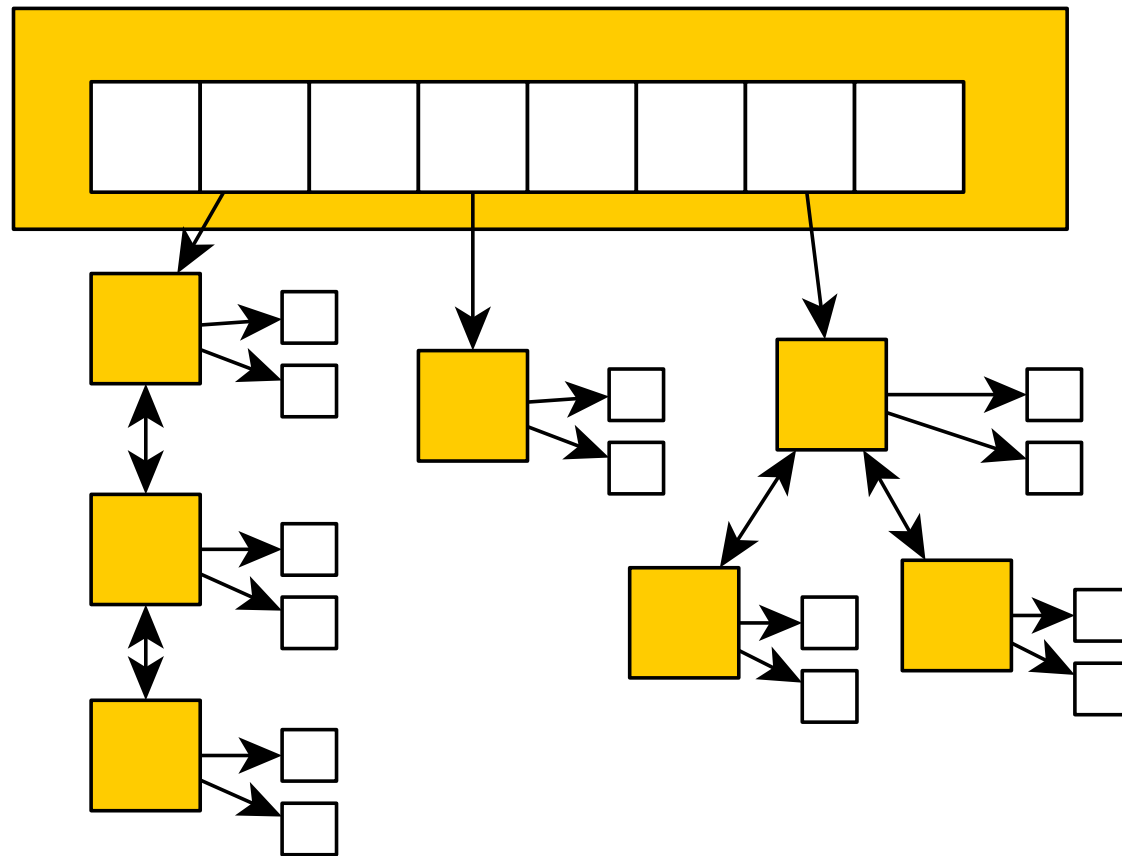
Структуры данных в Java



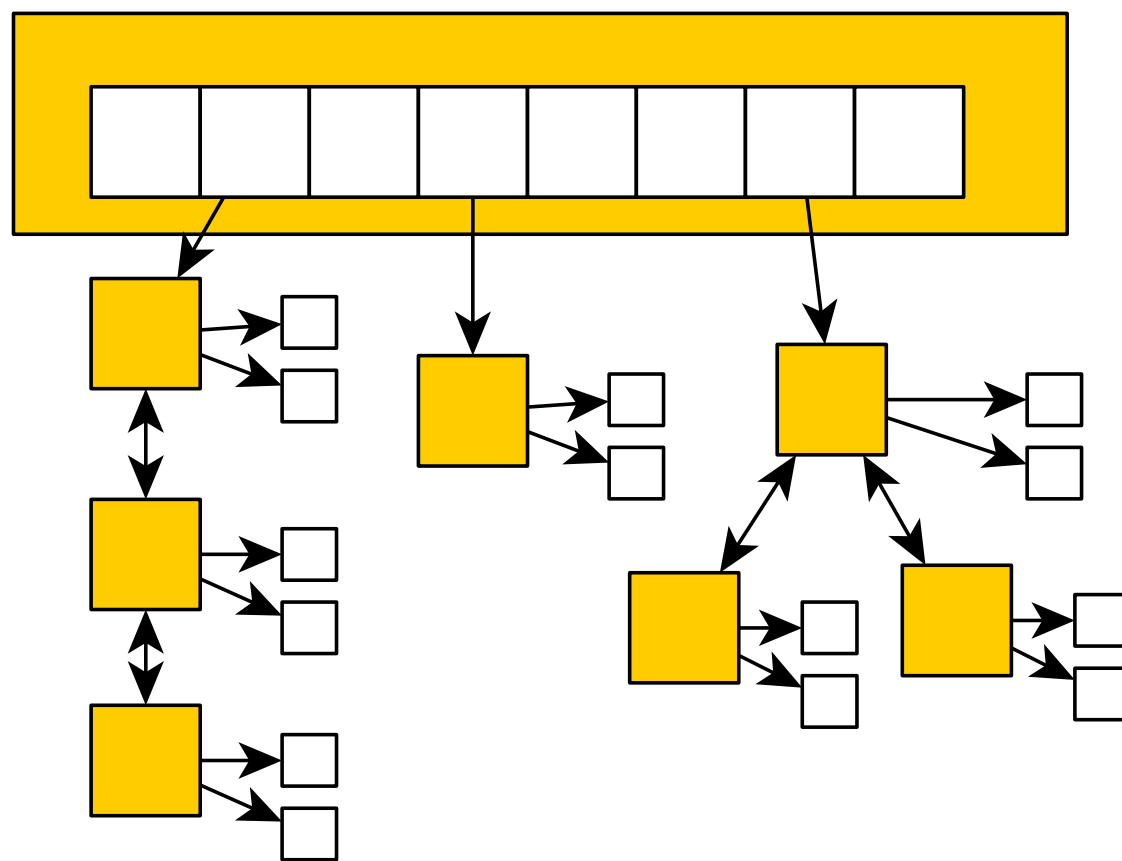
Структуры данных в Java



Структуры данных в Java

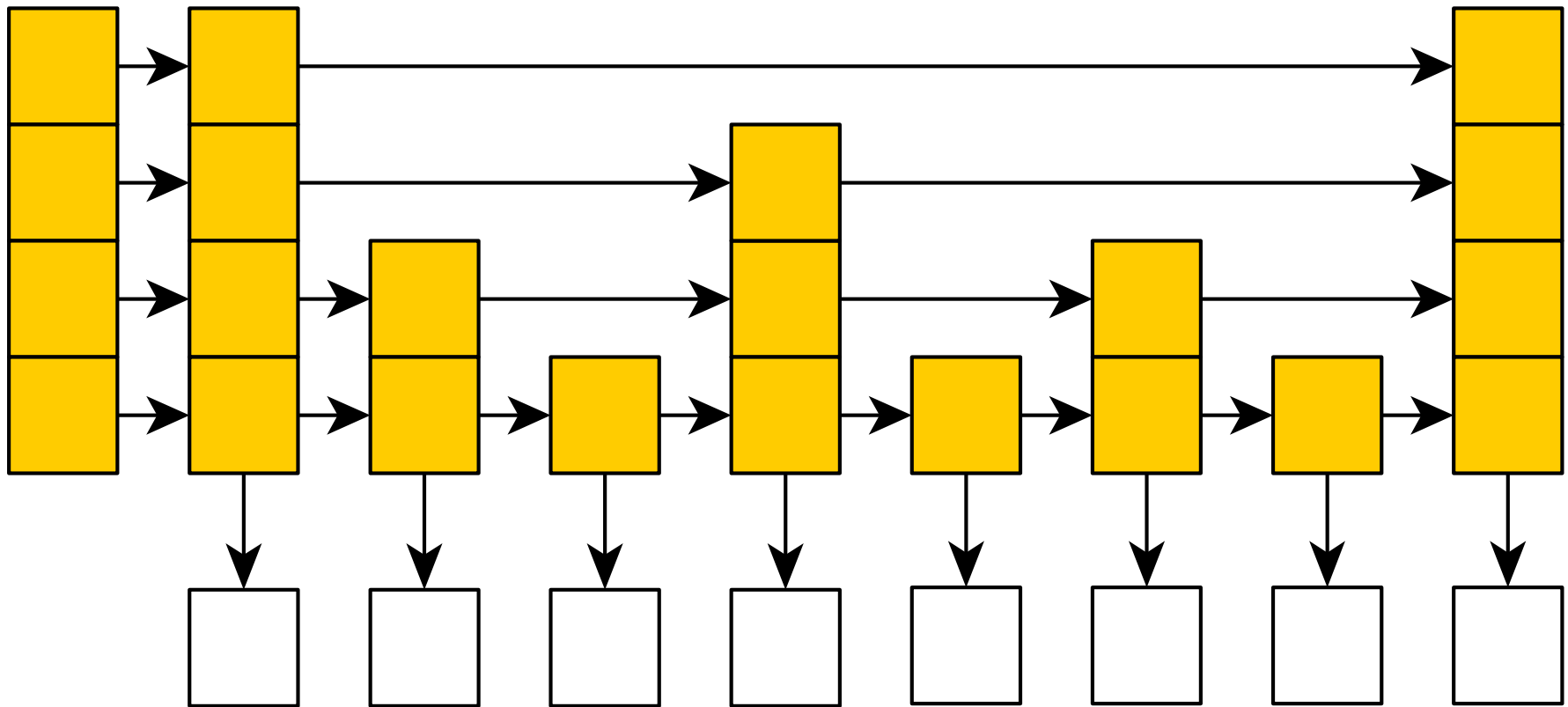


Структуры данных в Java

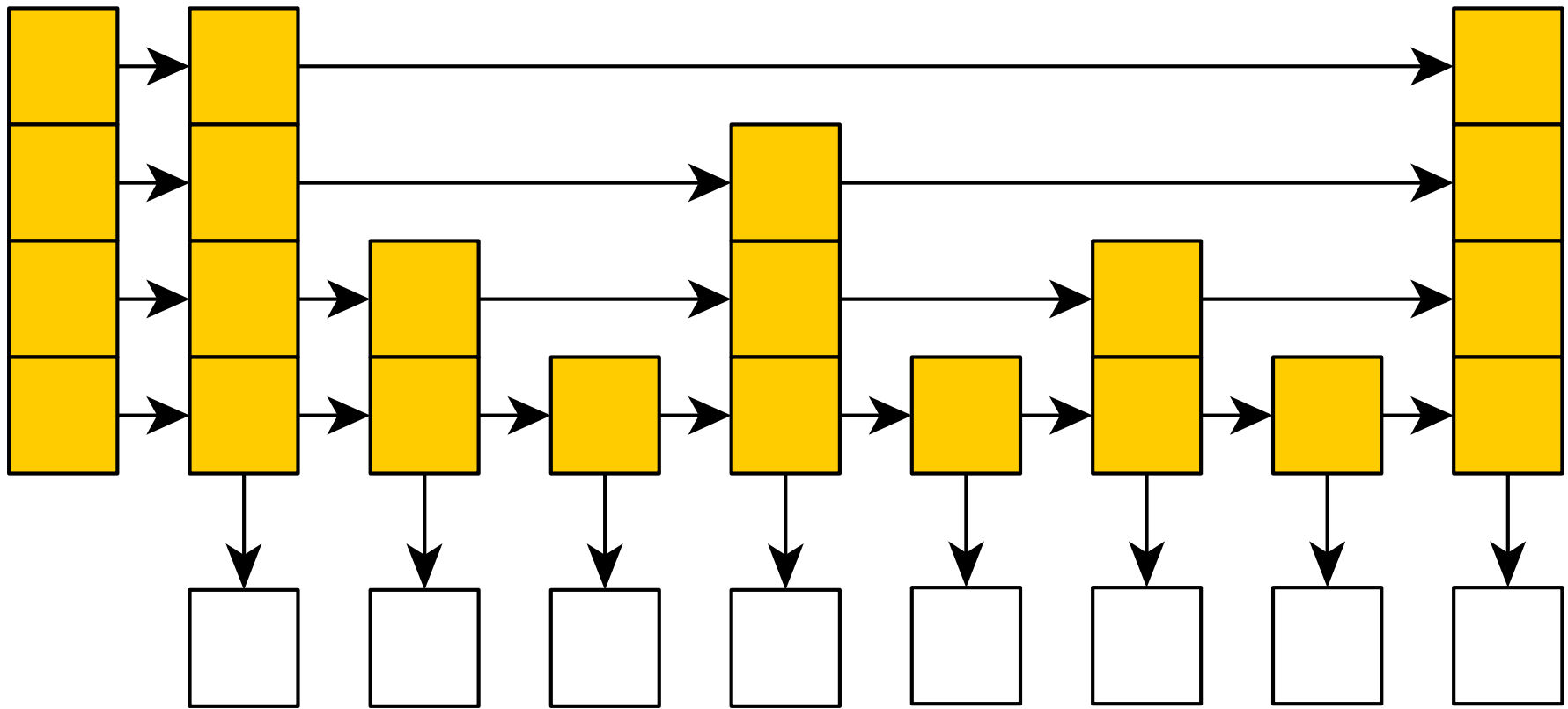


HashMap

Структуры данных в Java

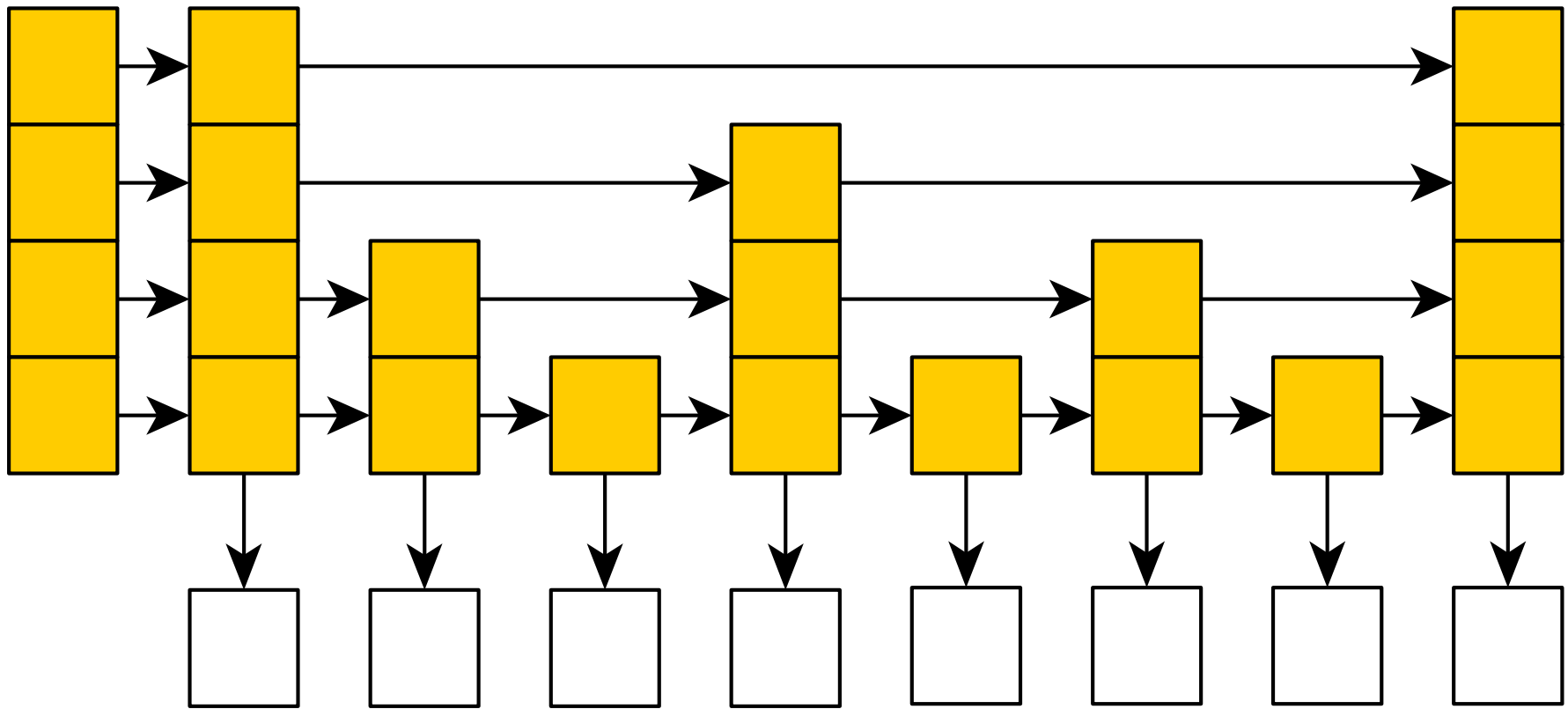


Структуры данных в Java



ConcurrentSkipListSet

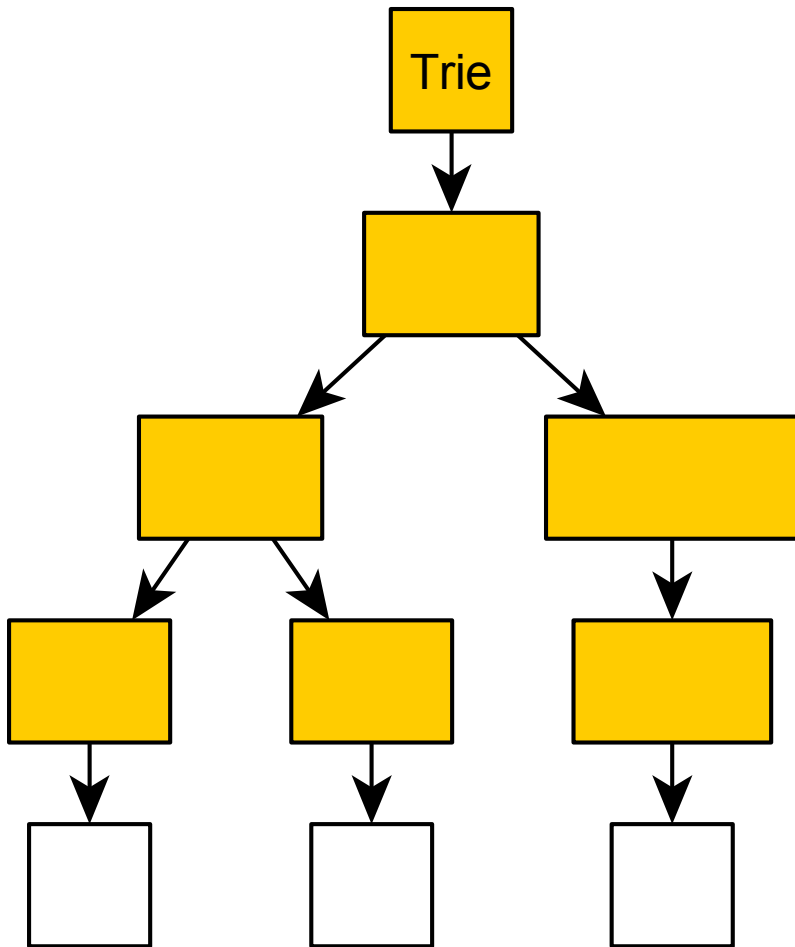
Структуры данных в Java



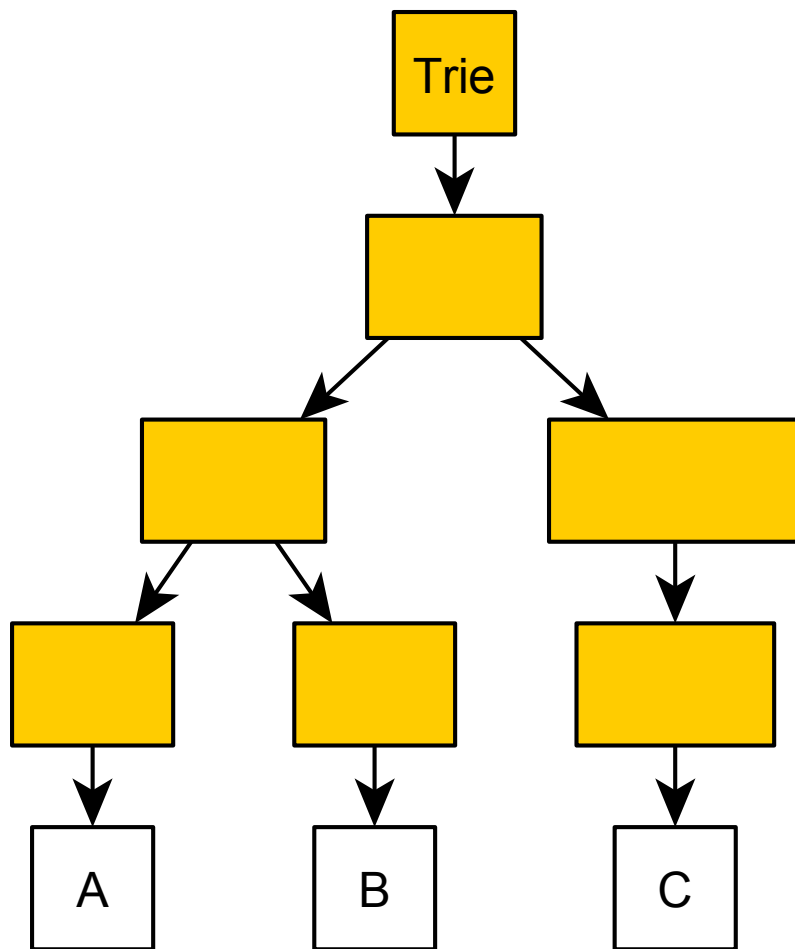
ConcurrentSkipListSet

Trie

- Trie – дерево

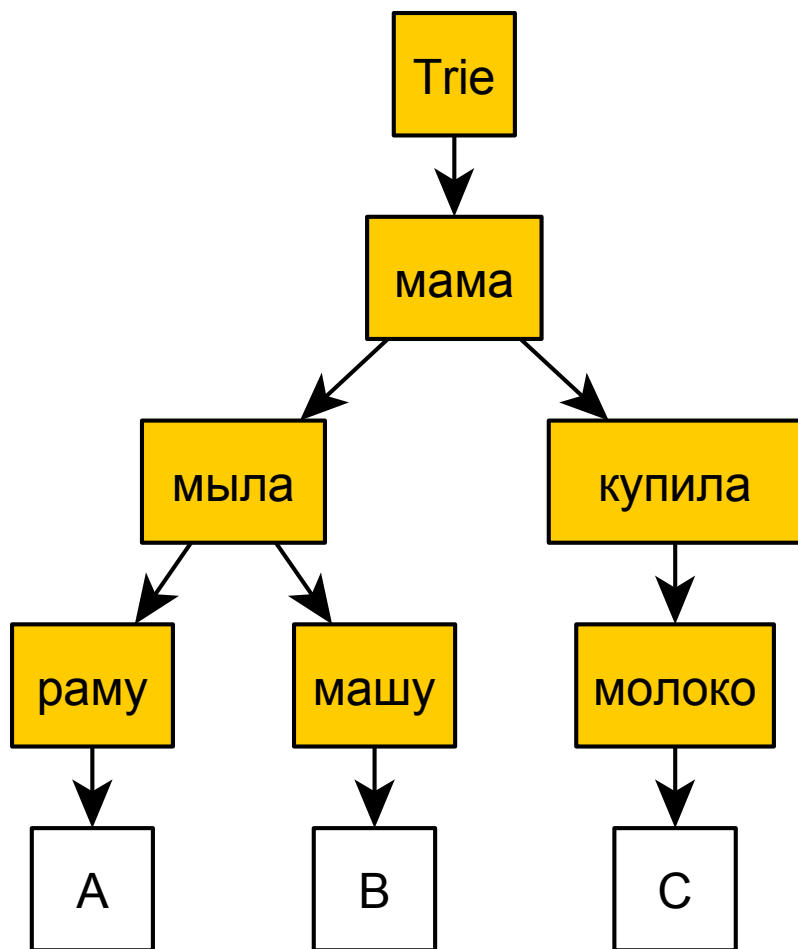


Trie



- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:
 - (ключ₁) → A
 - (ключ₂) → B
 - (ключ₃) → C

Trie



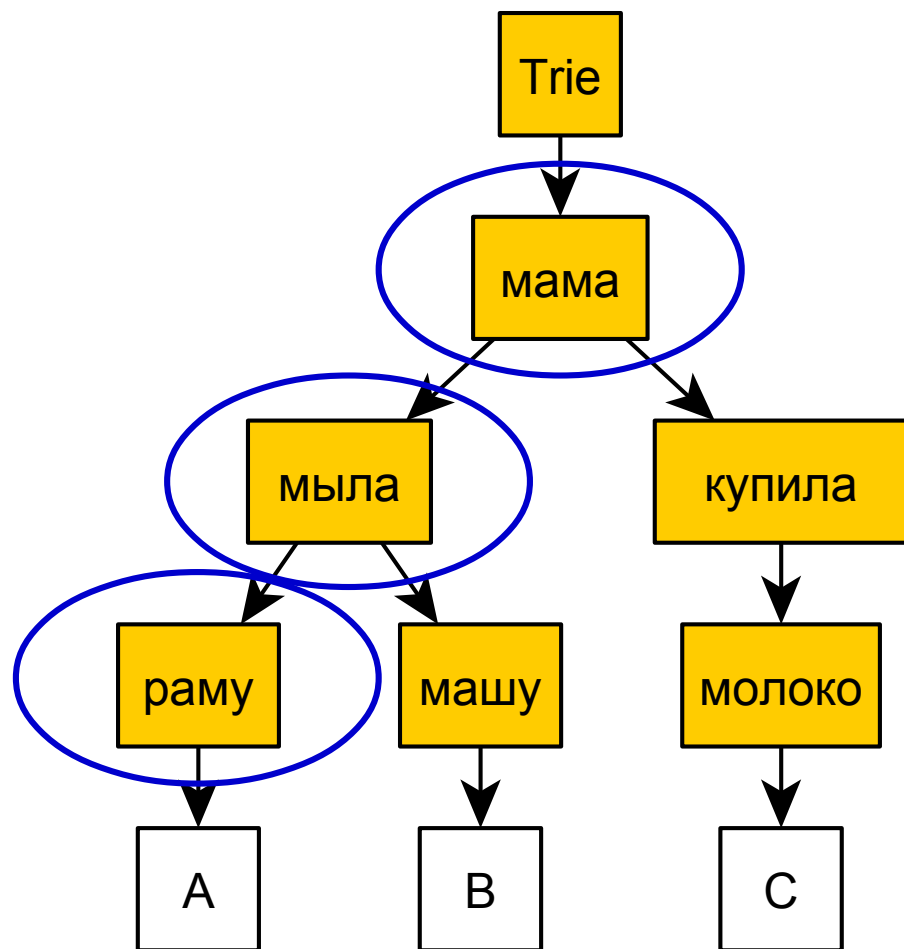
- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → А

«мама мыла машу» → В

«мама купила молоко» → С

Trie



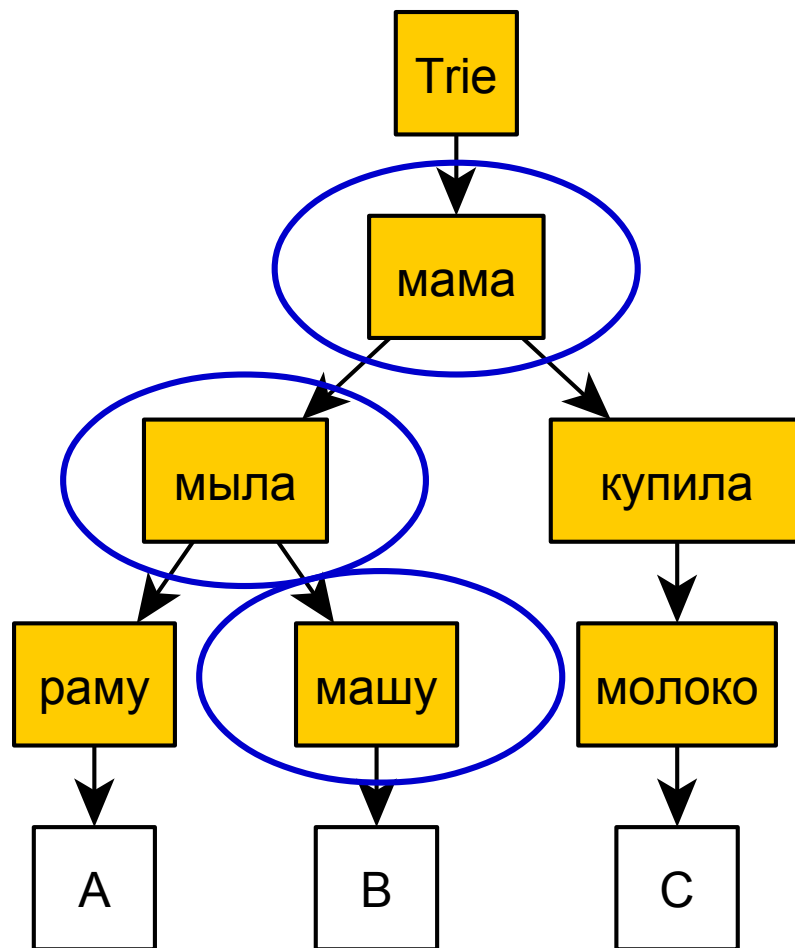
- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → А

«мама мыла машу» → В

«мама купила молоко» → С

Trie



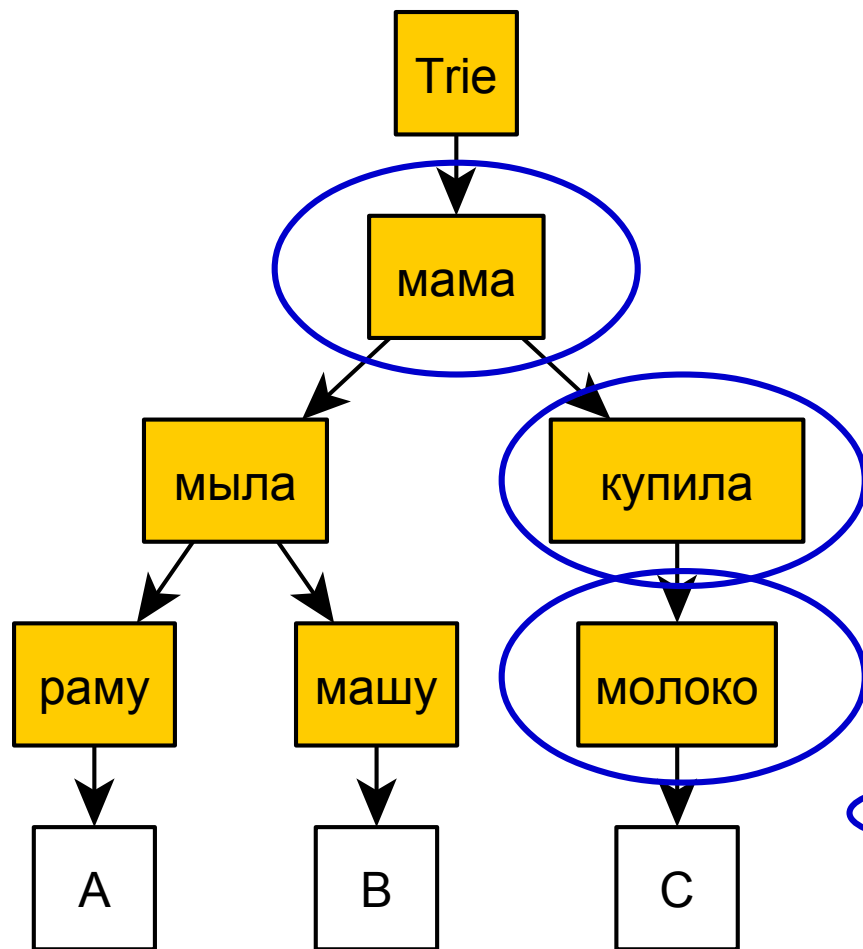
- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → А

«мама мыла машу» → В

«мама купила молоко» → С

Trie



- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → A

«мама мыла машу» → B

«мама купила молоко» → C

Trie: применения

- Хранение "суффиксных деревьев"
 - когда ключи имеют общие префиксы
- регулярные выражения
- "нечёткий" поиск
- См. например:
 - *Jetty-9 goes fast with Mechanical Sympathy. Look ahead Trie.*

<https://webtide.com/jetty-9-goes-fast-with-mechanical-sympathy/>

Trie

```
class Trie<V> {  
    Node<V> root = new Node<>();  
}  
  
class Node<V> {  
    Map<String, Node<V>> children = new HashMap<>();  
    String keyPart;  
    V value;  
}
```

Рейтинг авторов русской Википедии

WIKIPEDIA
The Free Encyclopedia



Рейтинг авторов русской Википедии

- Есть много разных рейтингов:
 - количество начатых статей ("первая правка")
 - количество правок в статьях
 - "стаж"

Рейтинг авторов русской Википедии

- Есть много разных рейтингов:
 - количество начатых статей ("первая правка")
 - количество правок в статьях
 - "стаж"
- Не отражают "качество правки"
- Не отражают популярность статей

Рейтинг авторов русской Википедии

- Построить рейтинг самых читаемых авторов русской Википедии в 2013 году

Рейтинг авторов русской Википедии

- Построить рейтинг самых читаемых авторов русской Википедии в 2013 году
 - пропорционально авторству всего текста в статье, а не факта создания статей

Рейтинг авторов русской Википедии

- Построить рейтинг самых читаемых авторов русской Википедии в 2013 году
 - пропорционально авторству всего текста в статье, а не факта создания статей
 - 1 балл за каждое посещение статьи в 2013 году ("хит") если ты 100% автор

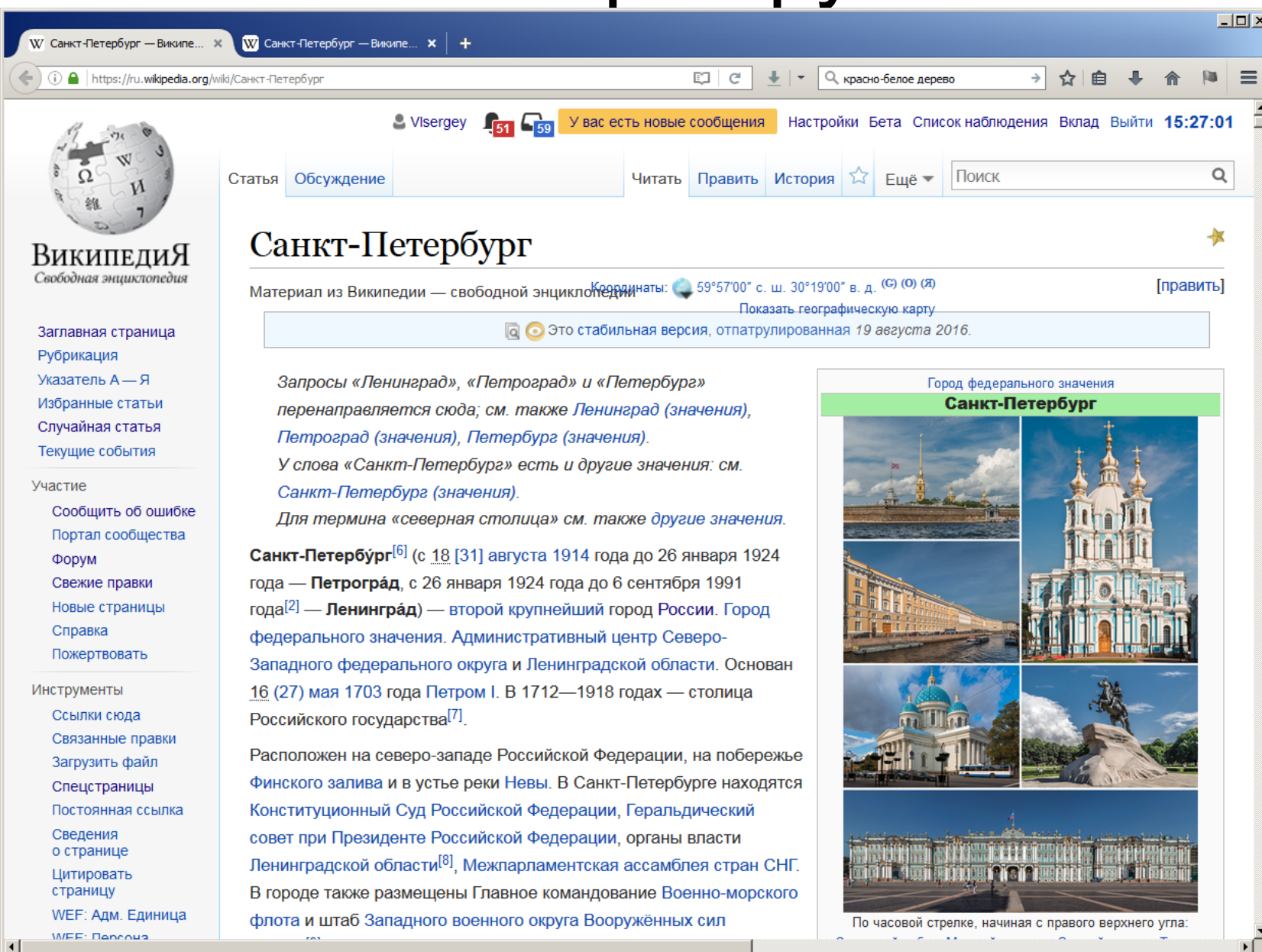
Рейтинг авторов русской Википедии

1. Забираем с stats.wikimedia.org годовую статистику посещений
2. Строим список первых N ($N > 1000$) популярных статей
3. Для каждой статьи определяем авторство в процентах
4. Считаем итоги

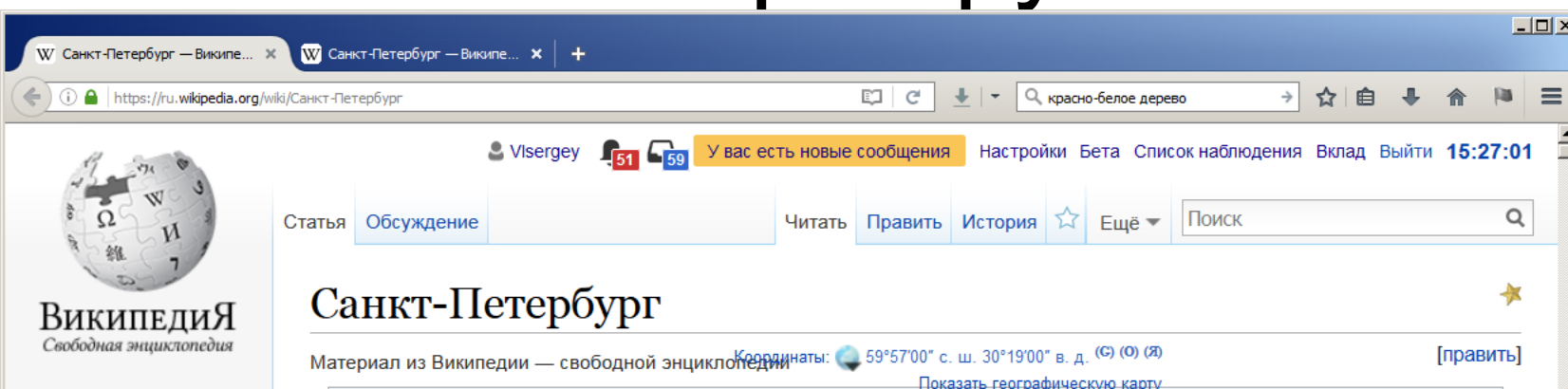
Рейтинг авторов русской Википедии

1. Забираем с stats.wikimedia.org годовую статистику посещений
2. Строим список первых N ($N > 1000$) популярных статей
3. Для каждой статьи определяем авторство в процентах
4. Считаем итоги

Рейтинг авторов русской Википедии



Рейтинг авторов русской Википедии



W Санкт-Петербург — Википедия

https://ru.wikipedia.org/wiki/Санкт-Петербург

У вас есть новые сообщения

Настройки Бета Список наблюдения Вклад Выйти 15:27:01

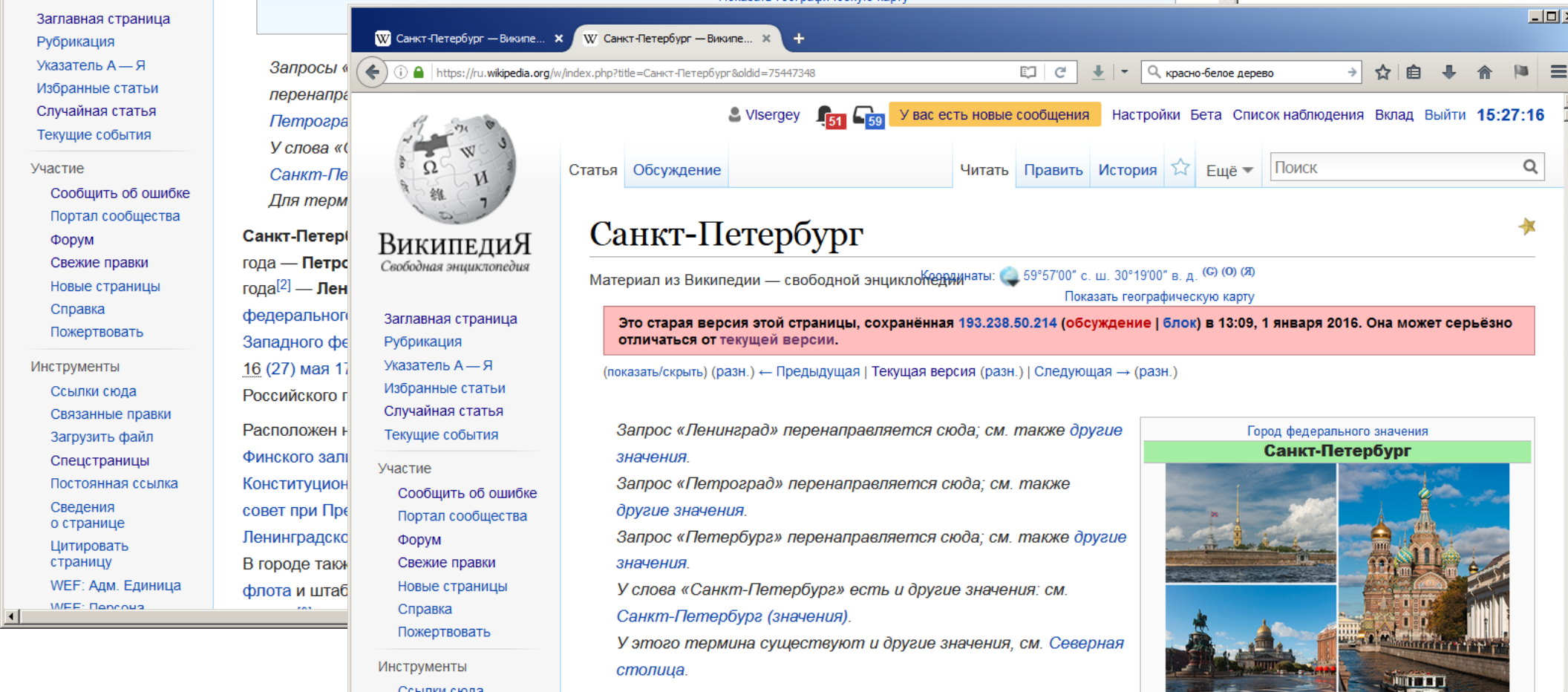
Статья Обсуждение Читать Править История Ещё Поиск

Санкт-Петербург

Материал из Википедии — свободной энциклопедии

Координаты: 59°57′00″ с. ш. 30°19′00″ в. д.﻿(???) [править]

[Показать географическую карту](#)



W Санкт-Петербург — Википедия

https://ru.wikipedia.org/w/index.php?title=Санкт-Петербург&oldid=75447348

У вас есть новые сообщения

Настройки Бета Список наблюдения Вклад Выйти 15:27:16

Статья Обсуждение Читать Править История Ещё Поиск

Санкт-Петербург

Материал из Википедии — свободной энциклопедии

Координаты: 59°57′00″ с. ш. 30°19′00″ в. д.﻿(???) [править]

[Показать географическую карту](#)

Это старая версия этой страницы, сохранённая 193.238.50.214 (обсуждение | блок) в 13:09, 1 января 2016. Она может серьёзно отличаться от текущей версии.

(показать/скрыть) (разн.) ← Предыдущая | Текущая версия (разн.) | Следующая → (разн.)

Запрос «Ленинград» перенаправляется сюда; см. также другие значения.

Запрос «Петроград» перенаправляется сюда; см. также другие значения.

Запрос «Петербург» перенаправляется сюда; см. также другие значения.

У слова «Санкт-Петербург» есть и другие значения: см. Санкт-Петербург (значения).

У этого термина существуют и другие значения, см. Северная столица.

Заглавная страница
Рубрикация
Указатель А — Я
Избранные статьи
Случайная статья
Текущие события

Участие

Сообщить об ошибке
Портал сообщества
Форум
Свежие правки
Новые страницы
Справка
Пожертвовать

Инструменты

Ссылки сюда
Связанные правки
Загрузить файл
Спецстраницы
Постоянная ссылка
Сведения о странице
Цитировать страницу
WTF: Адм. Единица
WTF: Персона

Санкт-Петербург

года — **Петр**

года^[2] — **Лен**

федерально

Западного фе

16 (27) мая 17

Российского г

Расположен н

Финского зали

Конституцион

совет при Пре

Ленинградско

В городе такж

флота и штаб

Заглавная страница
Рубрикация
Указатель А — Я
Избранные статьи
Случайная статья
Текущие события

Участие

Сообщить об ошибке
Портал сообщества
Форум
Свежие правки
Новые страницы
Справка
Пожертвовать

Инструменты

Ссылки сюда

Город федерального значения

Санкт-Петербург



Рейтинг авторов русской Википедии



Участие

Инструменты

Atom

У вас есть новые сообщения

Настройки Бета Список наблюдения Вклад Выйти 15:28:59

Обсуждение

[Читать](#)

Правиль

История

Ewe ▾

Поиск

 [Помощь](#)

[Показать журналы для этой страницы](#) · [Скрытие правок](#) · [Настройки стабилизации](#) · [Вернуть к...](#)

[Просмотреть историю](#)

С года (и ранее): 2016

С месяца (и ранее):

Фильтр меток:

Только скрытые

Показать

Это страница истории изменений. Внешние инструменты: [статистика правок](#) • [поиск правки](#) • [статистика посещений](#). См. также [сведения о странице](#), [список подстраниц](#), [срабатывания фильтров](#).

([новейшие](#) | [старейшие](#)) [Просмотреть \(250 более новых | 250 более старых\) \(20 | 50 | 100 | 250 | 500\)](#)

Сравнить выбранные версии

Выбор: Все, Ничего, Инвертировать

- (текущ. | пред.) 02:22, 18 августа 2016 92.100.191.58 (обсуждение | блок) . . (300 440 байтов) (+10) . . (→Радиостанции) (откатить 1 правку | отменить) (Метки: Правка с моб. устройства, Правка через мобильную версию сайта) [отпатрулирована участником AlexTref871]
- (текущ. | пред.) 21:37, 17 августа 2016 Повелитель Звёзд (обсуждение | вклад | блок) . . (300 430 байтов) (-139) . . (отмена правки 80284770 участника 37.214.235.76 (обс)к стаб. версии) (отменить | поблагодарить) [отпатрулирована участником Vs64vs]
- (текущ. | пред.) 21:33, 17 августа 2016 37.214.235.76 (обсуждение | блок) . . (300 569 байтов) (+137) . .

Рейтинг авторов русской Википедии

[https://ru.wikipedia.org/w/api.php](https://ru.wikipedia.org/w/api.php?action=query&prop=revisions&rvlimit=10&titles=Санкт-Петербург&rvprop=timestamp|user|comment&format=json)
?action=query
&prop=revisions
&rvlimit=10
&titles=Санкт-Петербург
&rvprop=timestamp|user|comment
&format=json

Определение авторства текста

- Взять самую первую версию статьи
(27 декабря 2002 года)
- Узнать его автора
(участник Theta682)
- Считать, что это автор 100% оригинального
текста статьи (на 27 декабря 2002 года)

Определение авторства текста

- Взять следующую версию статьи
(7 января 2003)
- Найти новые предложения и словосочетания
в тексте и пометить их авторство
- Считать, что автор новой версии имеет
столько авторства статьи, сколько нового
текста в ней

Сравнение двух текстов

- автор А:
- сгущённое молоко
— молоко вкусное
- автор Б:
- сгущённое молоко
— **очень** вкусное

Сравнение двух текстов

- построить рейтинг авторов
 - определить авторство текста
 - сравнить тексты между собой
 - ...

Сравнение двух текстов

- построить рейтинг авторов
 - определить авторство текста
 - сравнить тексты между собой
 - ...
 -
 -
 - Trie

Сравнение двух текстов

- построить рейтинг авторов
 - определить авторство текста
 - сравнить тексты между собой
 - алгоритм поиска наибольшей общей подстроки

Сравнение двух текстов

- Автор А:
- сгущённое молоко
— молоко вкусное
- Автор Б:
- сгущённое молоко
— **очень** вкусное
- 1. "сгущённое молоко" (автор А)

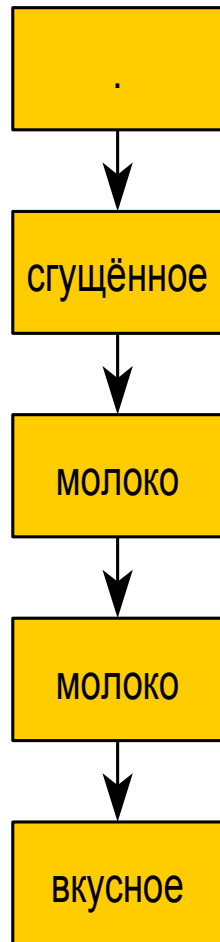
Сравнение двух текстов

- Автор А:
 - ~~сгущённое молоко~~
—молоко вкусное
 - Автор Б:
 - ~~сгущённое молоко~~
—**очень** вкусное
-
- 1. "сгущённое молоко" (автор А)
 - 2. "вкусное" (автор А)

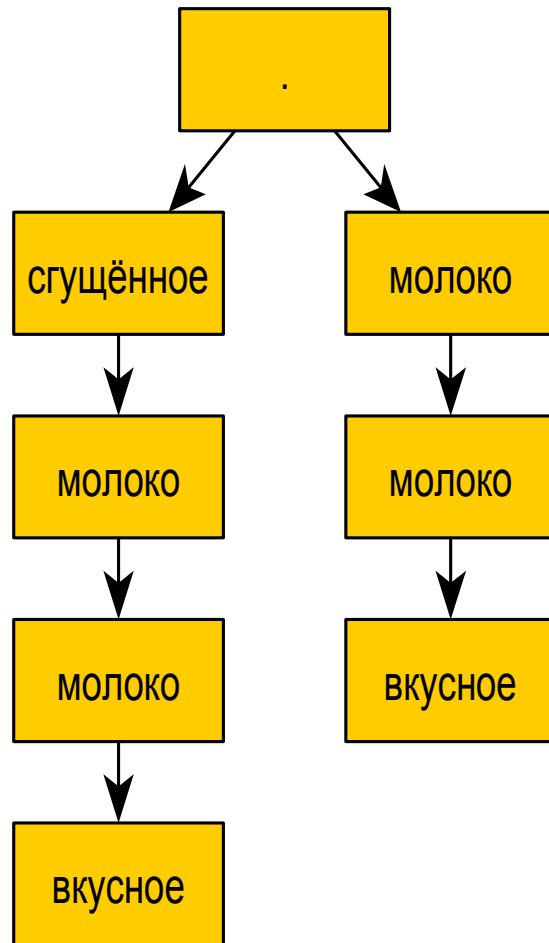
Сравнение двух текстов

- ~~Автор А:~~
 - ~~сгущённое молоко~~
— ~~молоко~~ ~~вкусное~~
 - Автор Б:
 - ~~сгущённое молоко~~
— **очень** ~~вкусное~~
-
- 1. "сгущённое молоко" (автор А)
 - 2. "вкусное" (автор А)
- ост.: "очень" (автор Б)

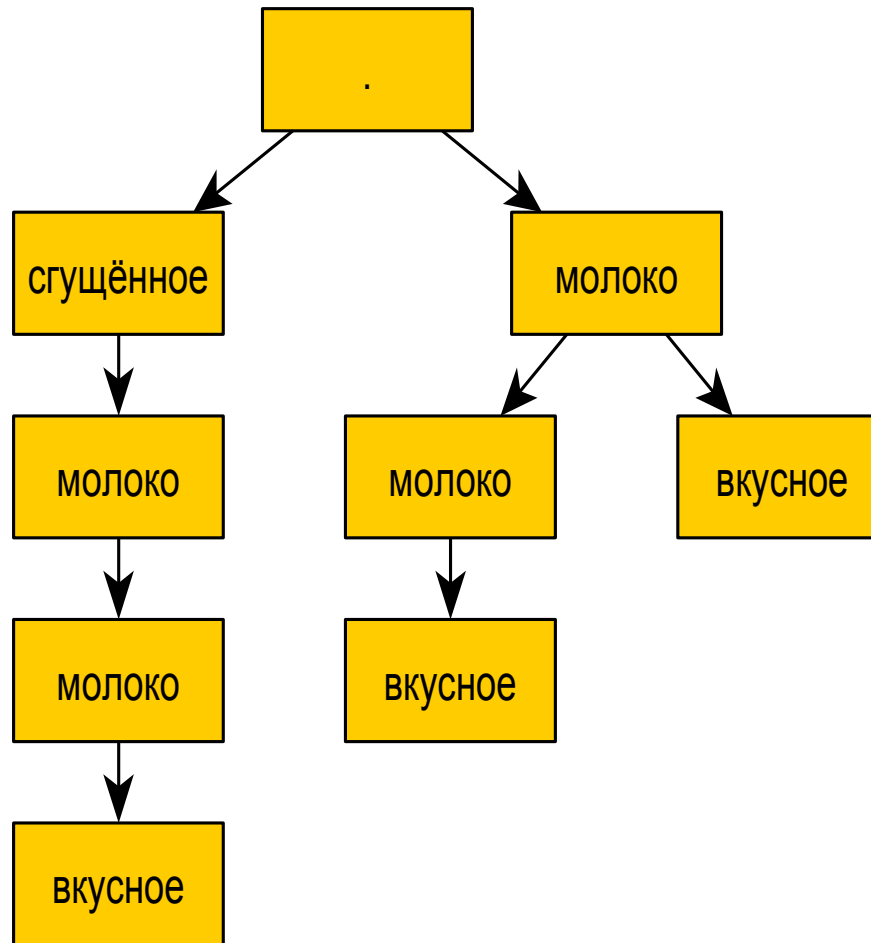
Поиск наибольшей общей подстроки



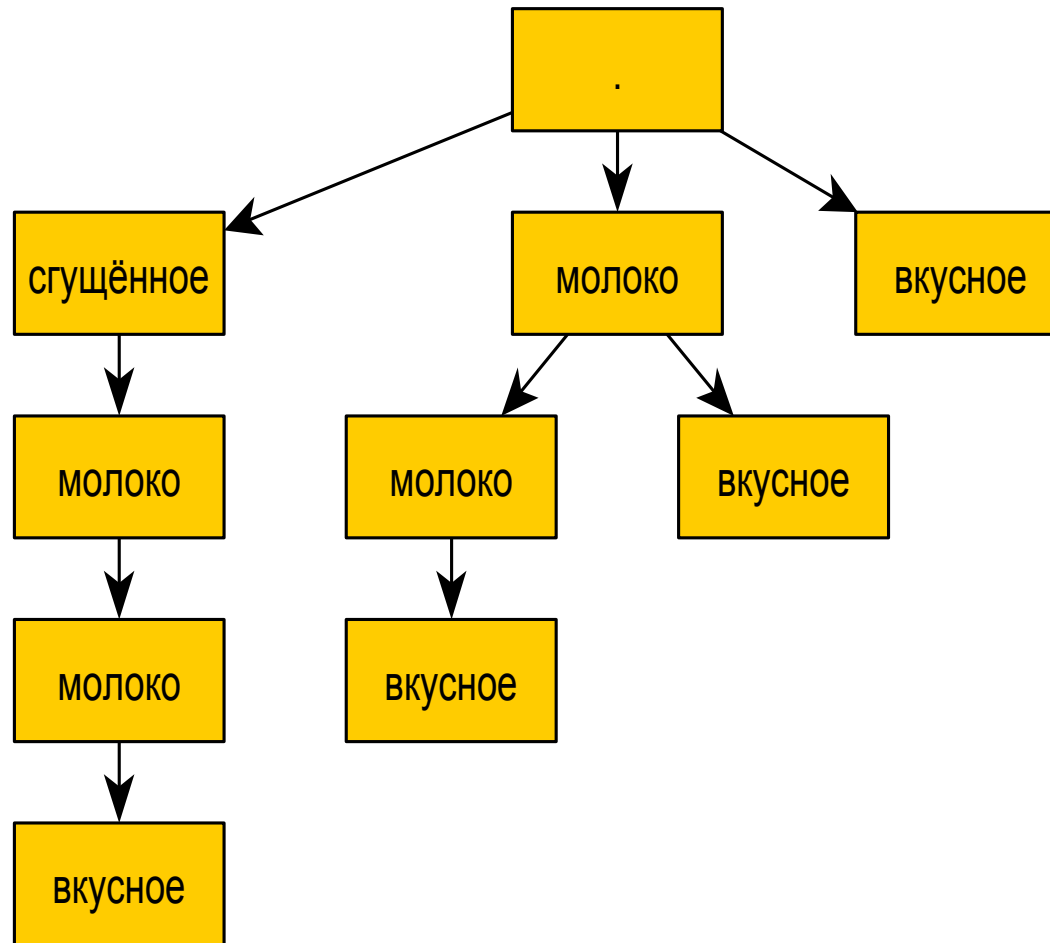
Поиск наибольшей общей подстроки



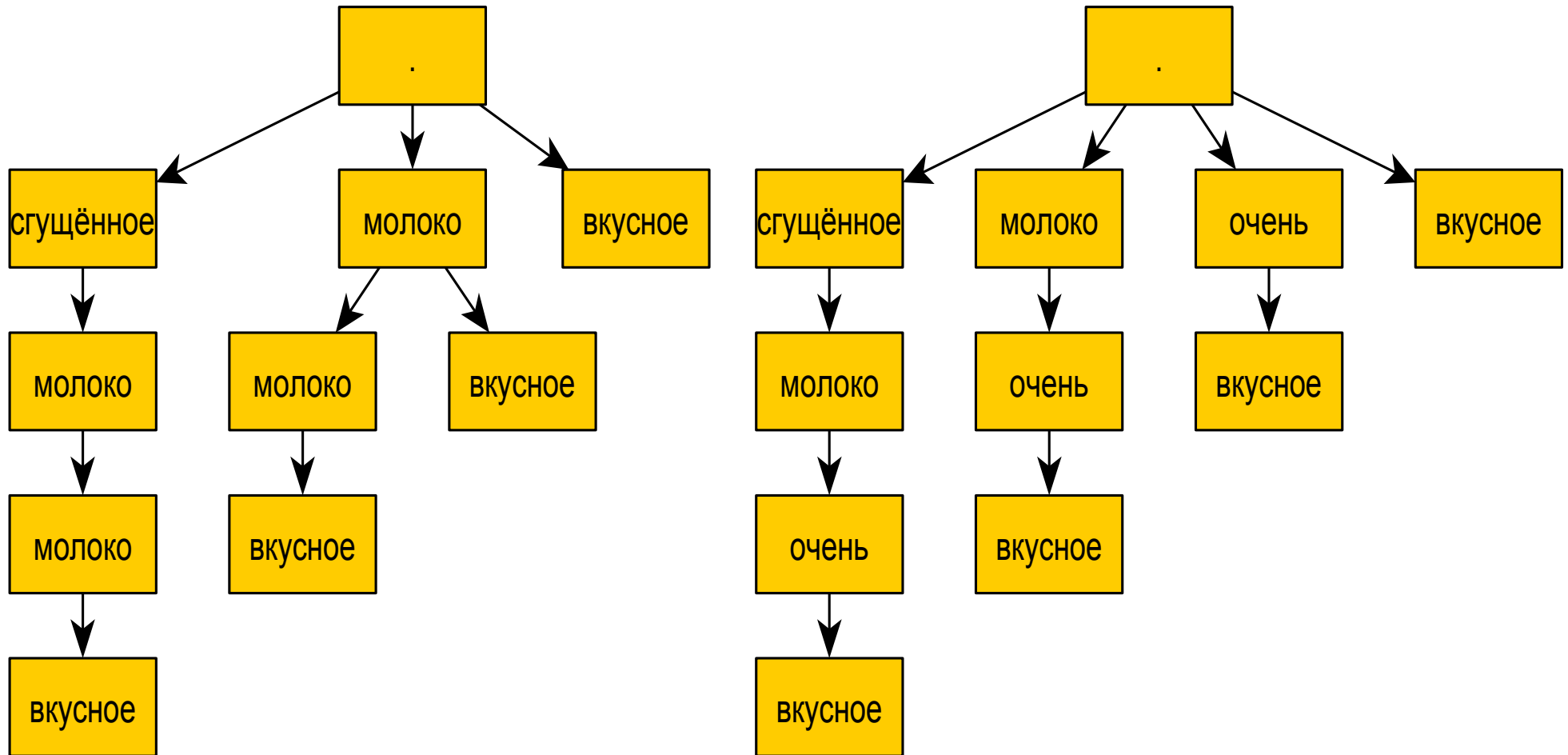
Поиск наибольшей общей подстроки



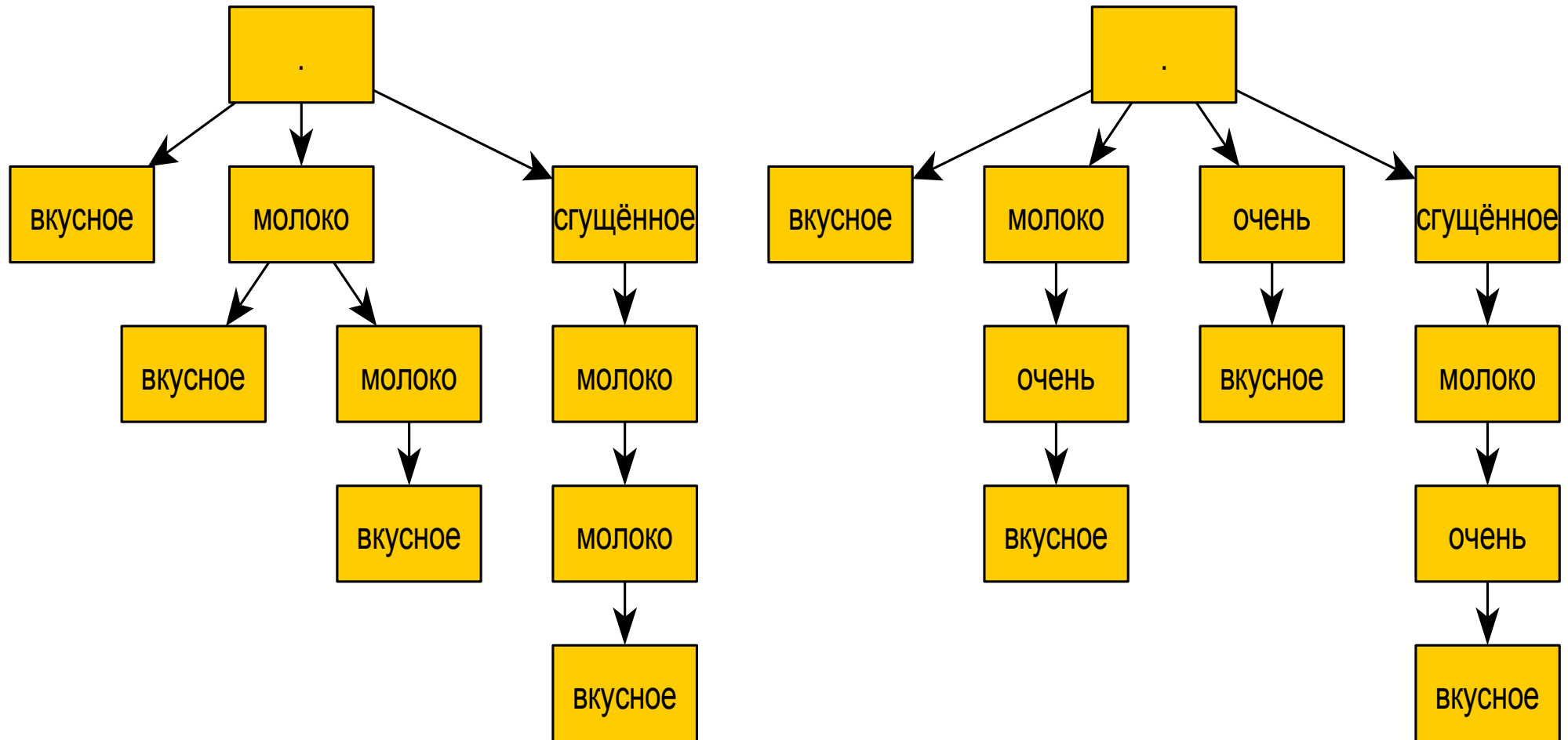
Поиск наибольшей общей подстроки



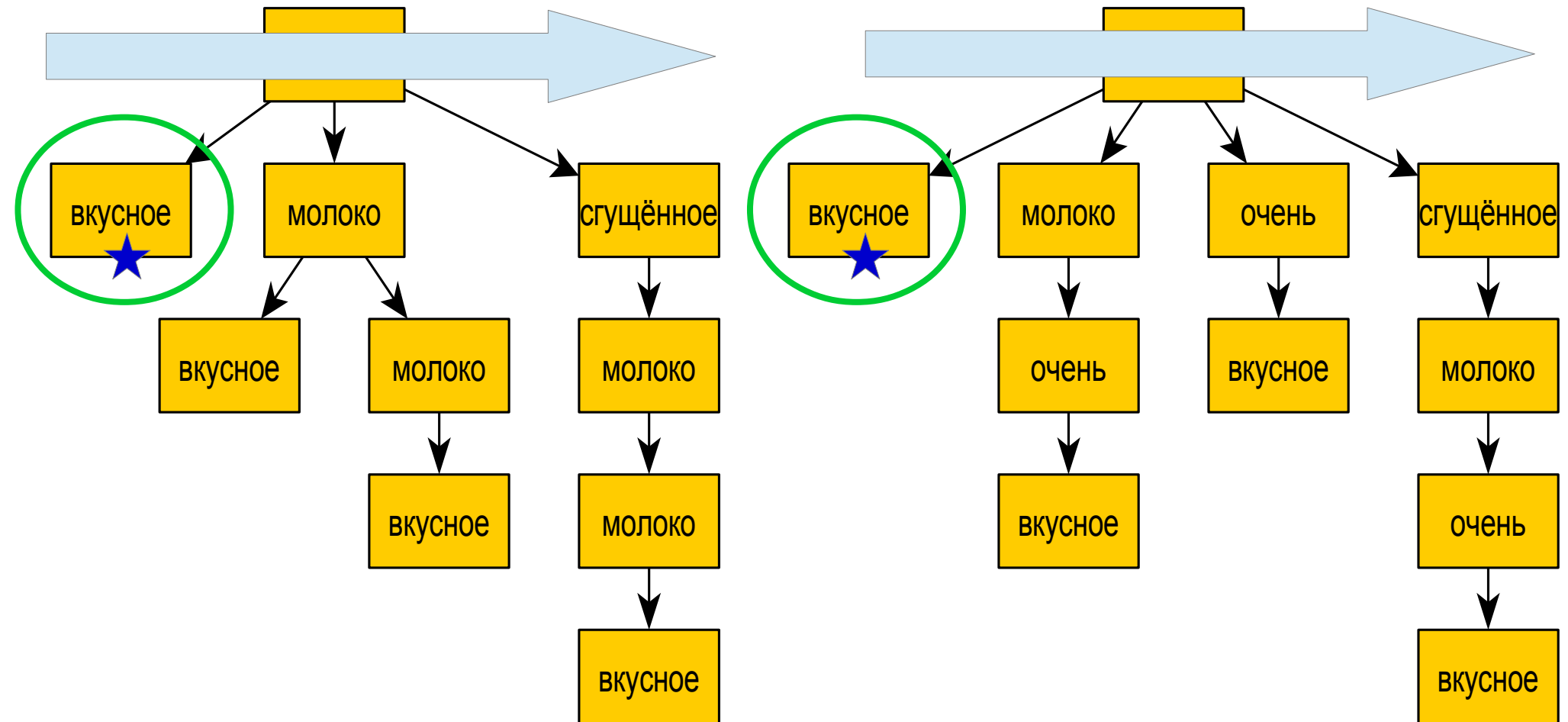
Поиск наибольшей общей подстроки



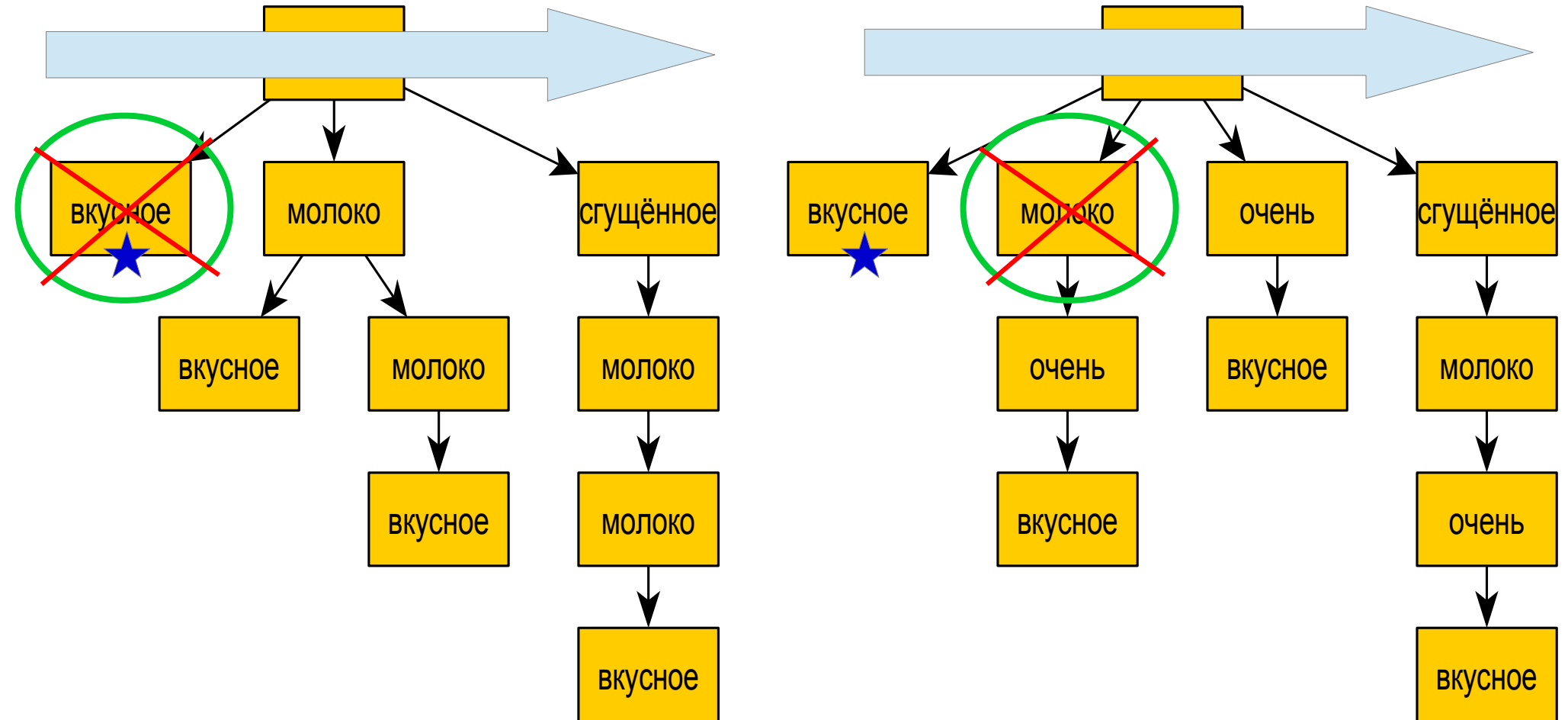
Поиск наибольшей общей подстроки



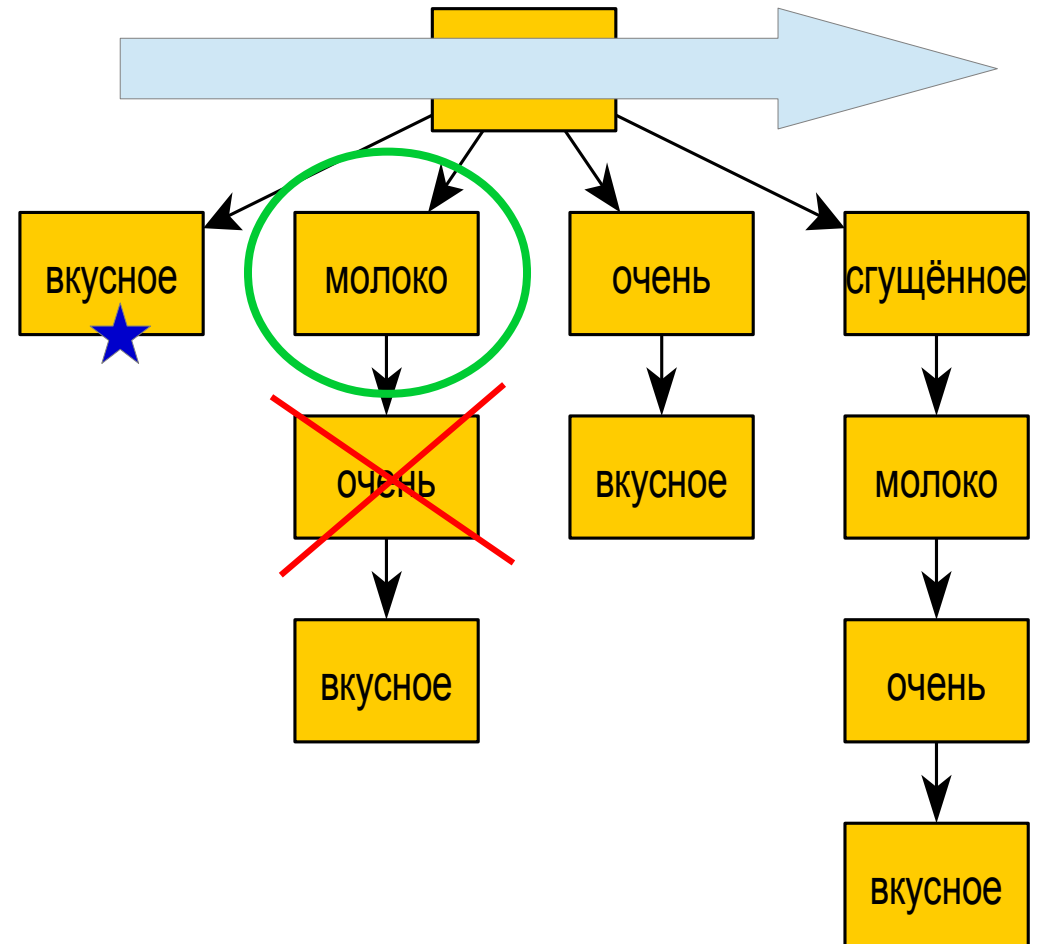
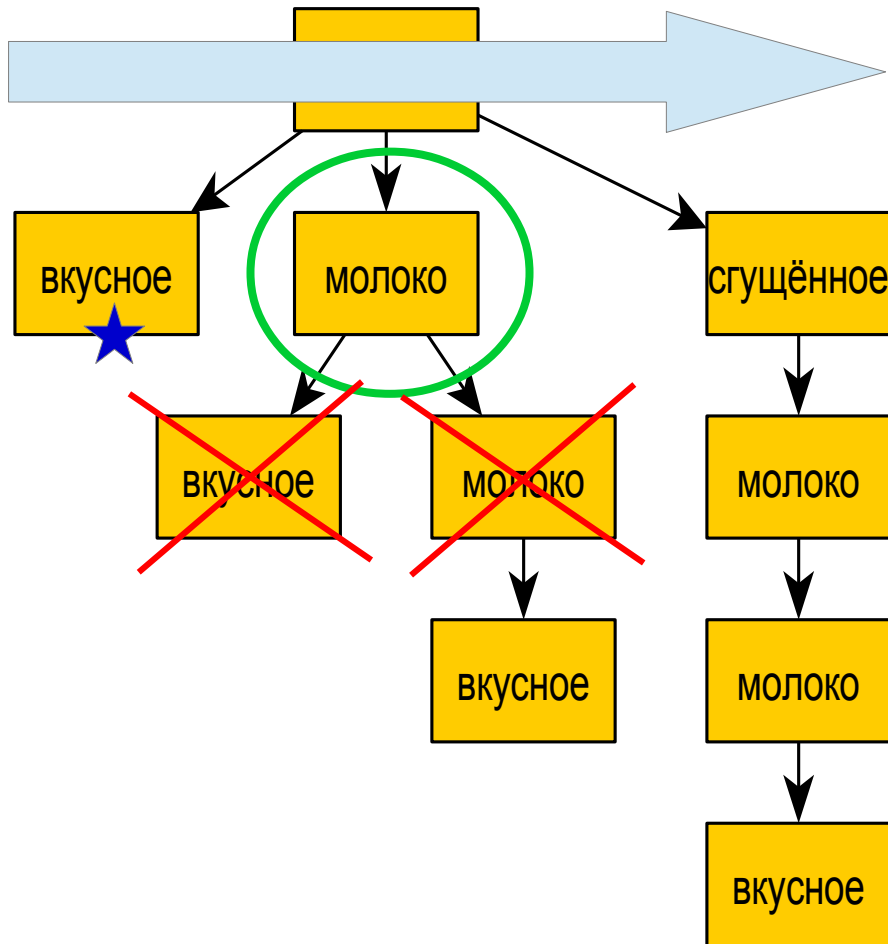
Поиск наибольшей общей подстроки



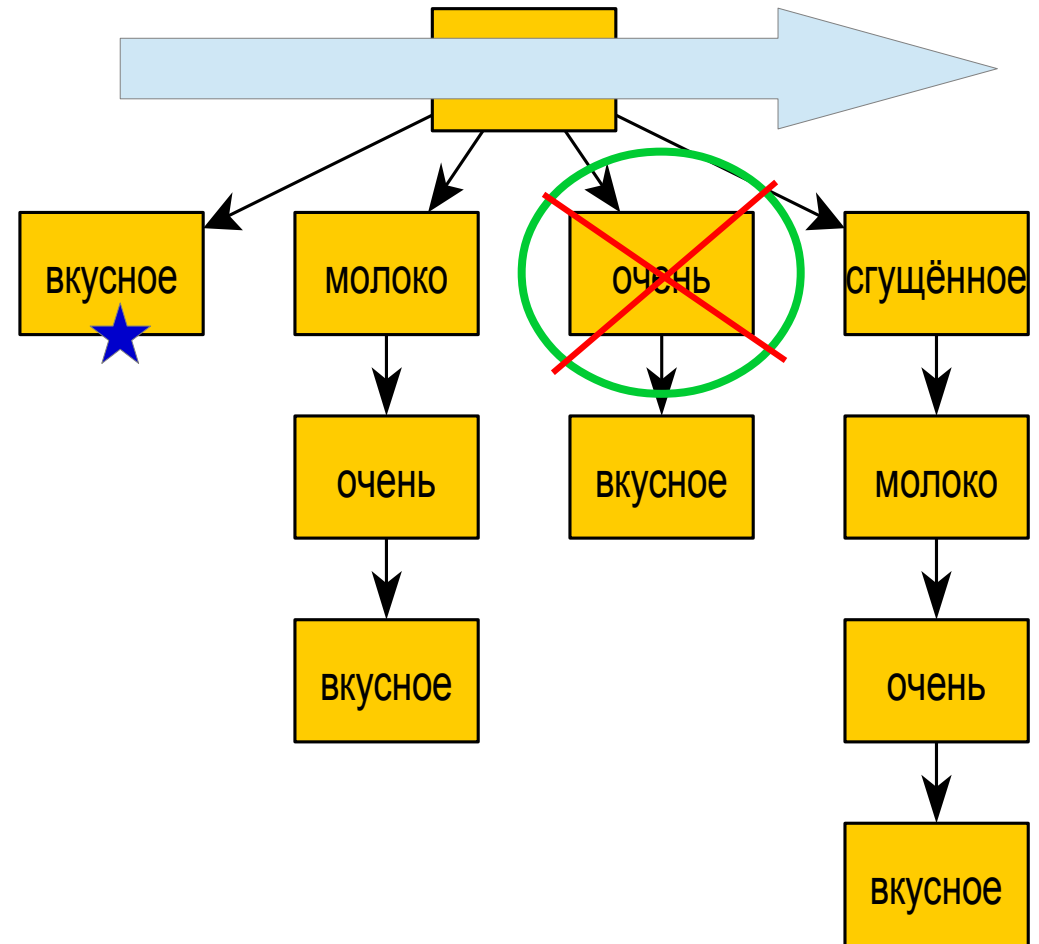
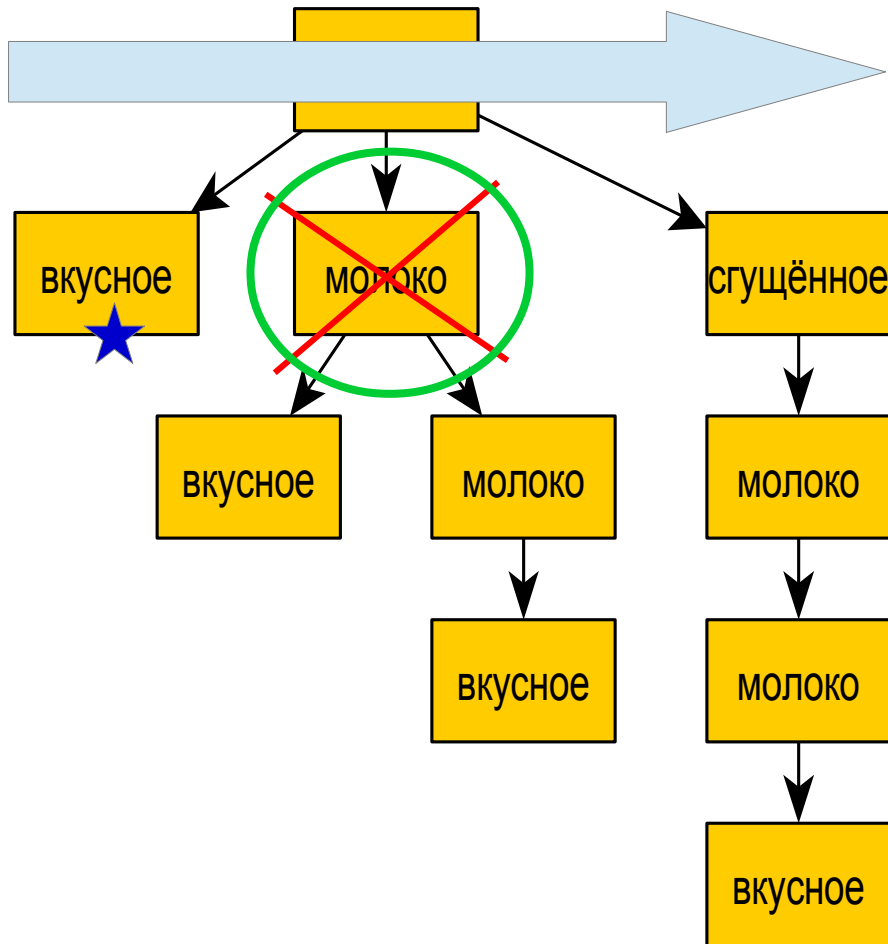
Поиск наибольшей общей подстроки



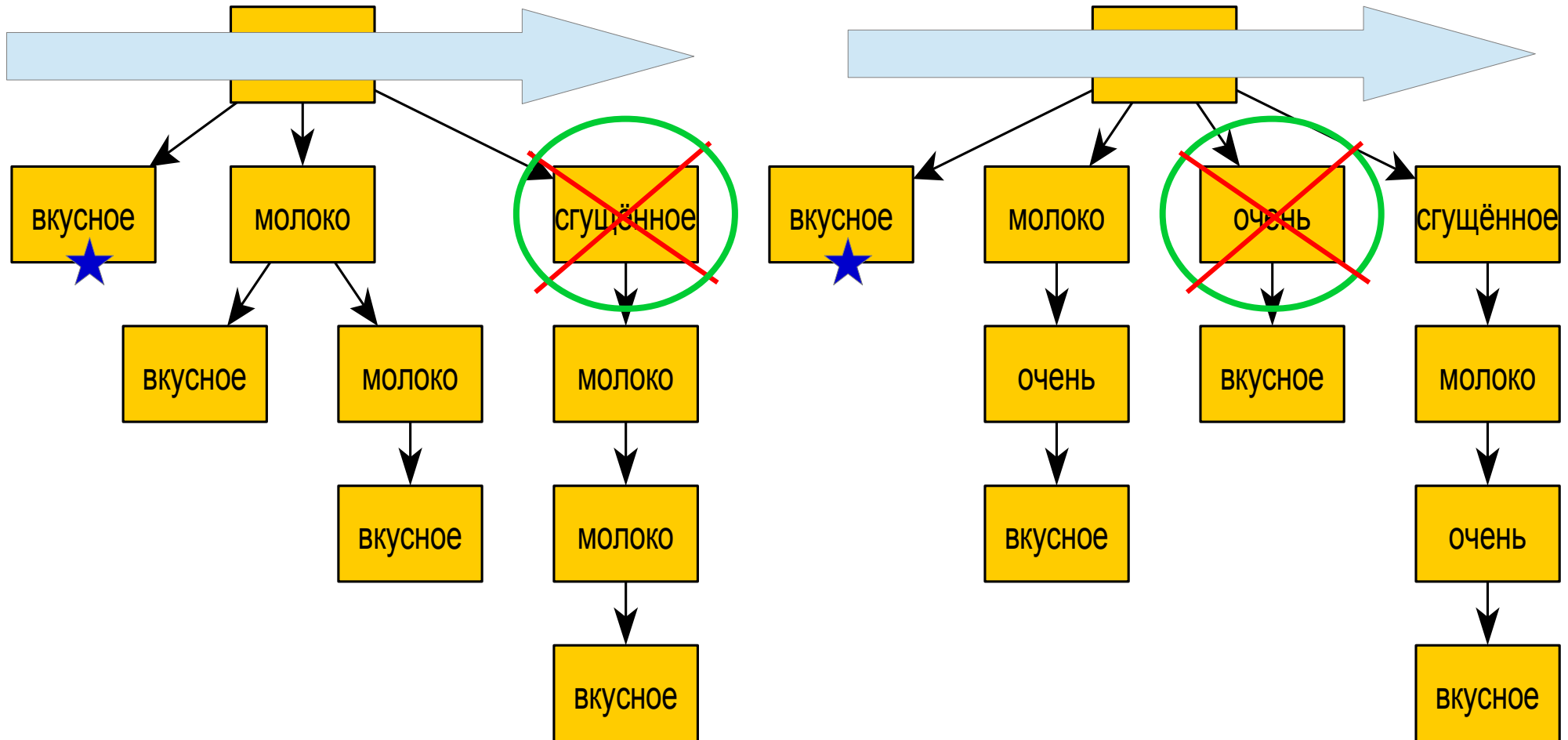
Поиск наибольшей общей подстроки



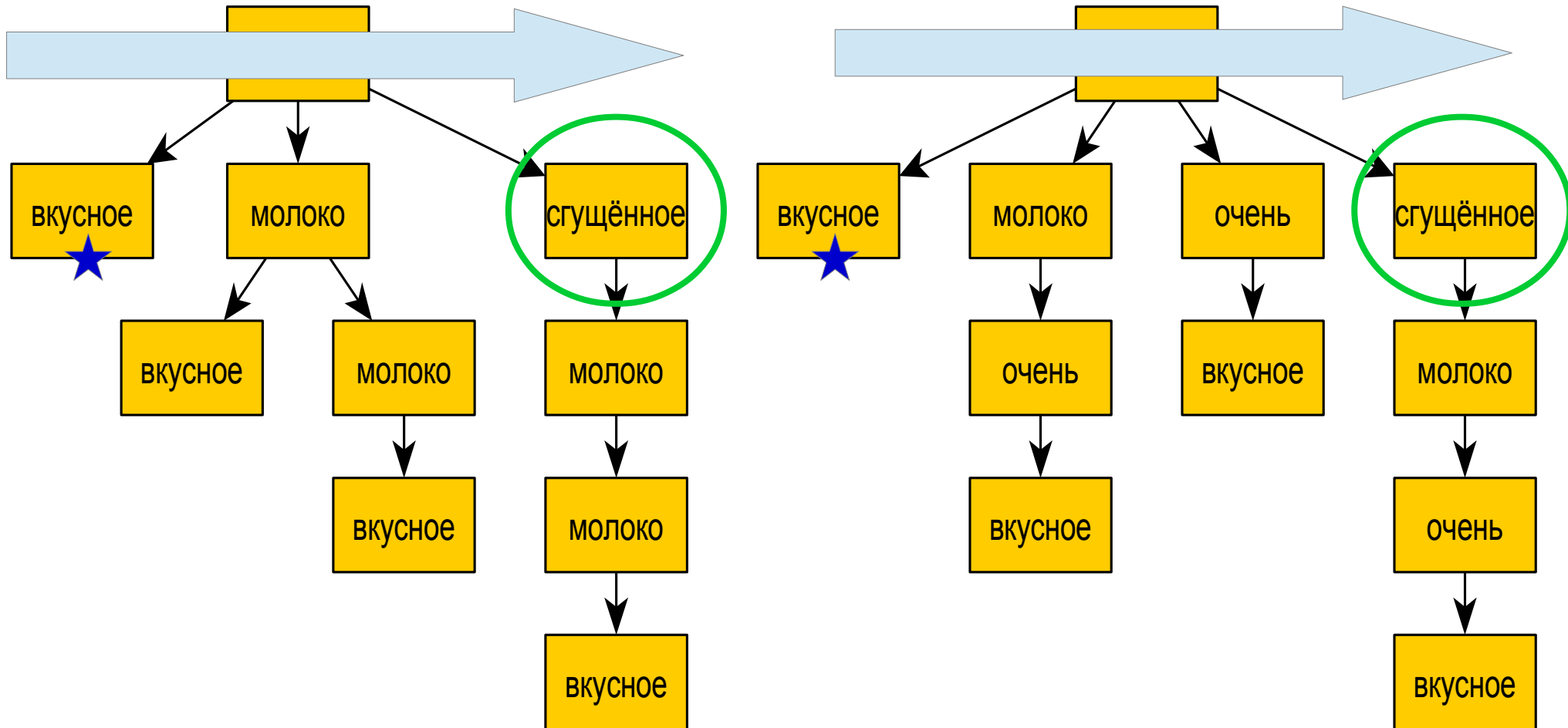
Поиск наибольшей общей подстроки



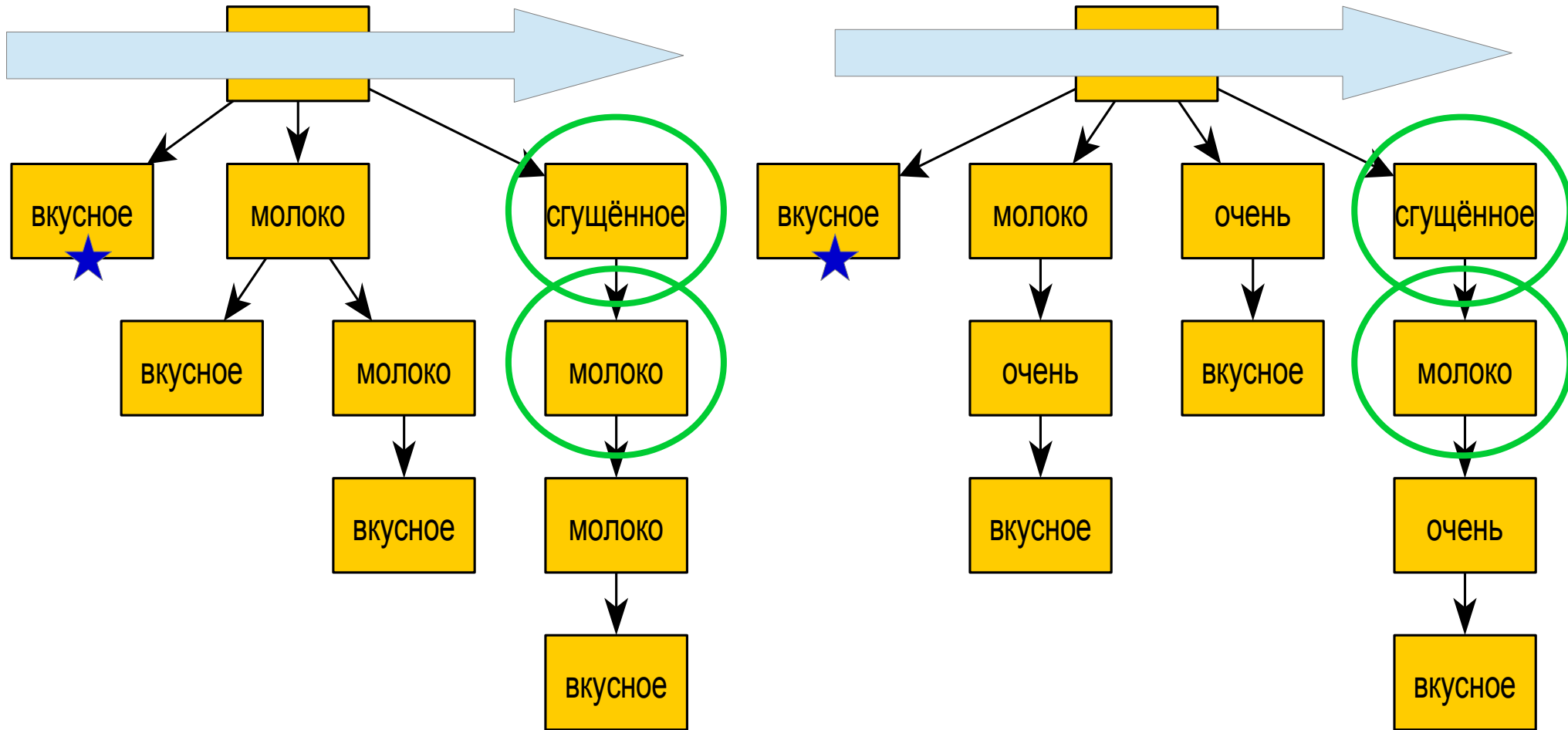
Поиск наибольшей общей подстроки



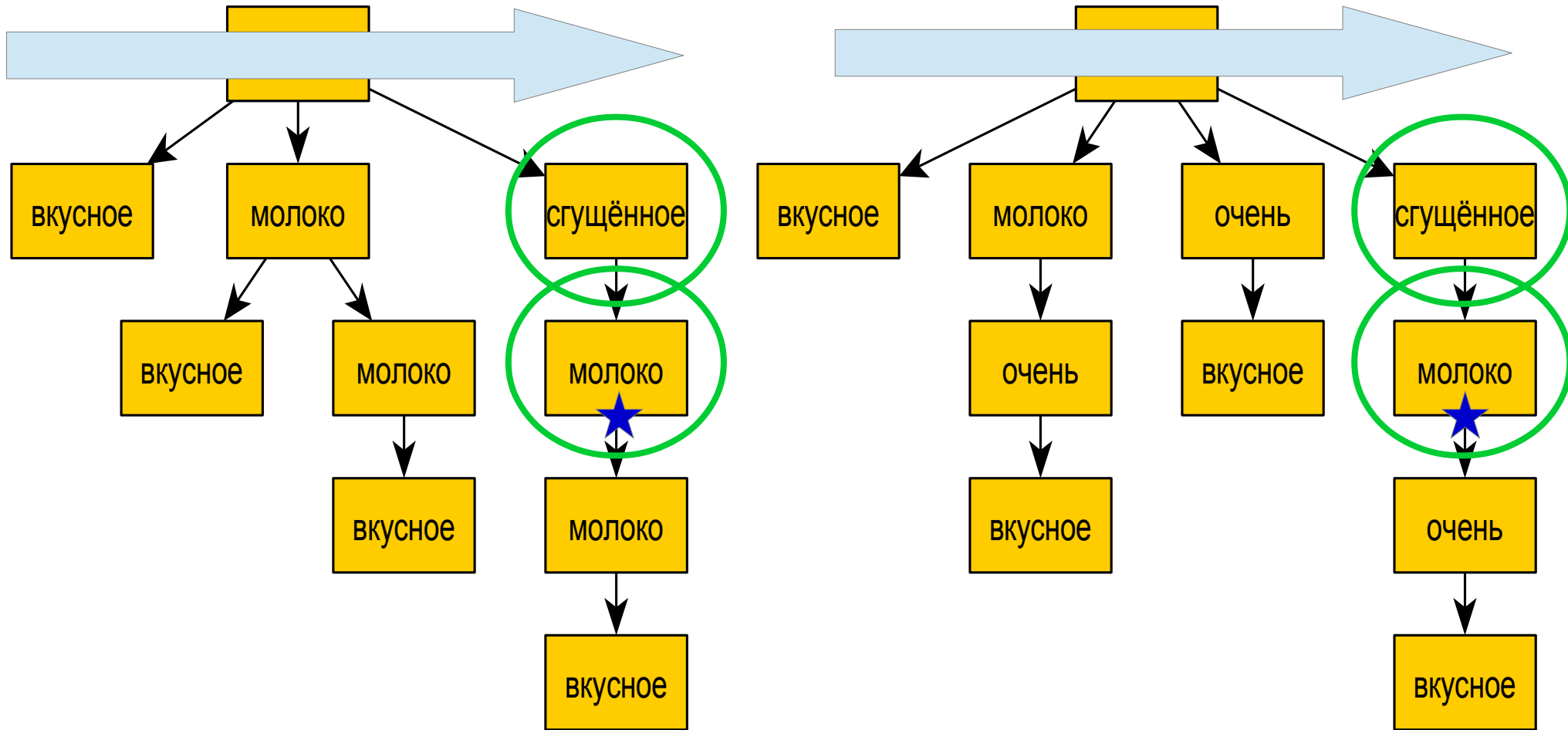
Поиск наибольшей общей подстроки



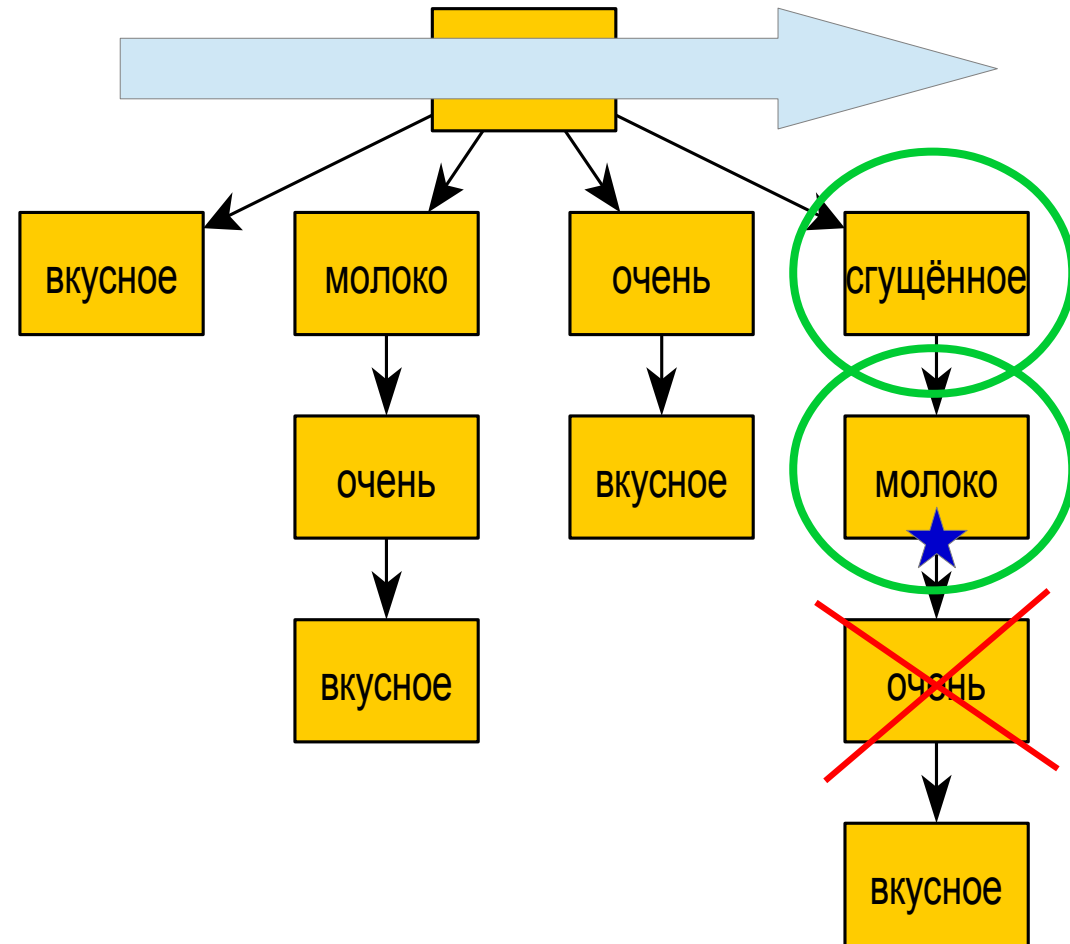
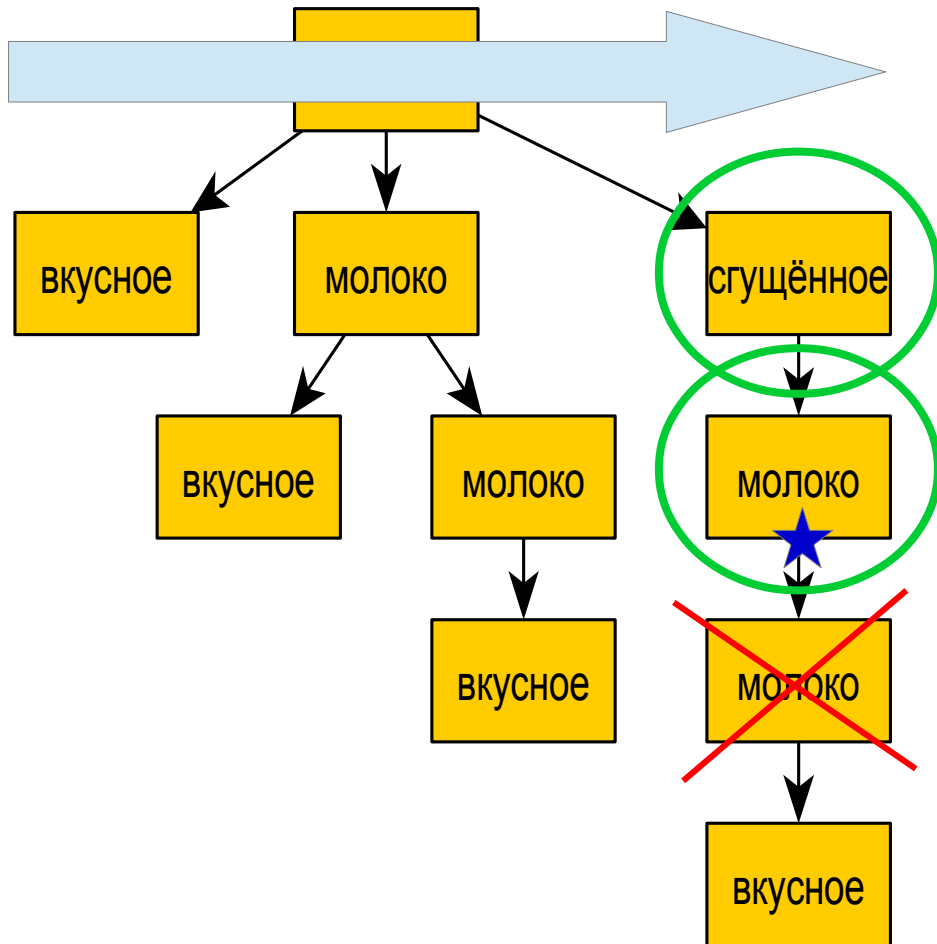
Поиск наибольшей общей подстроки



Поиск наибольшей общей подстроки



Поиск наибольшей общей подстроки



Повторять от забора и до...

- Взять первую версию статьи. Установить "метку" автору первой версии на каждое слово.

Повторять от забора и до...

- Взять первую версию статьи. Установить "метку" автору первой версии на каждое слово.
- Взять вторую версию статьи, найти общие строки с первой, вычеркнуть. Для оставшихся установить "метку" автора второй версии.

Повторять от забора и до...

- Взять первую версию статьи. Установить "метку" автору первой версии на каждое слово.
- Взять вторую версию статьи, найти общие строки с первой, вычеркнуть. Для оставшихся установить "метку" автора второй версии.
- Взять третью версию статьи...
- Взять четвёртую версию статьи...

Повторять от забора и до...

- После сравнения последней версии с предыдущей:
 - посчитать авторство текста по количеству меток на словах

Результат

- Мы смогли найти наибольшую подстроку в тексте

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии
- Мы смогли "начислить" авторам баллы

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии
- Мы смогли "начислить" авторам баллы
- Мы смогли построить рейтинг авторов русскоязычного раздела Википедии

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии
- Мы смогли "начислить" авторам баллы
- Мы смогли построить рейтинг авторов русскоязычного раздела Википедии
- и всё это за 2 недели 100% загрузки CPU, 50 Гб трафика и сна под вентилятор

Результат

Jul-Ar	60 177 баллов	1 355 статей
Valdis72	57 013 балла	3 148 статей
Ghirlandajo	50 702 балла	2 485 статей
Horim	39 765 баллов	969 статей
Alrofficial	36 036 баллов	1 357 статей

[//ru.wikipedia.org/wiki/Y:VlsergeyBot/WikiRaiting/2013](http://ru.wikipedia.org/wiki/Y:VlsergeyBot/WikiRaiting/2013)

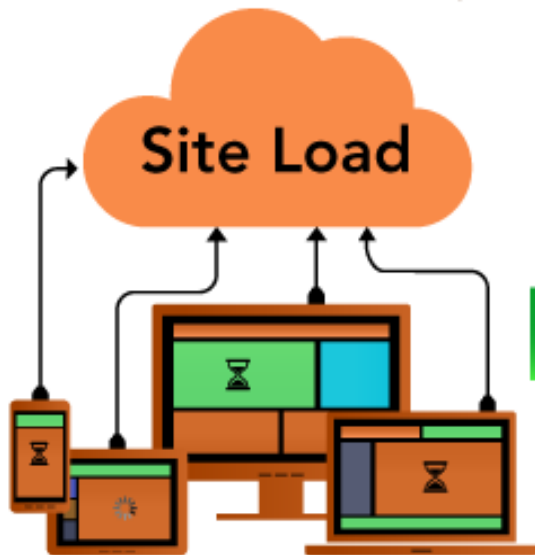
Результат

- Люди Икс: Дни минувшего будущего (2 080) — 89,49% (1 861)
- Росوماха: Бессмертный (2 225) — 64,37% (1 432)
- G.I. Joe: Бросок кобры 2 (1 180) — 67,06% (791)
- 300 спартанцев: Расцвет империи (788) — 96,82% (763)
- Человек из стали (1 850) — 41,12% (761)
- Оставшиеся статьи (1 350):
 - Средний вклад: 9,76% ± 21,21%
 - Посещений: 593 ± 736
 - Баллов за статью: 40 ± 93
 - Итого за оставшиеся: 53 704

RTB — real time bidding

1

Web user begins to load a webpage with an ad spot



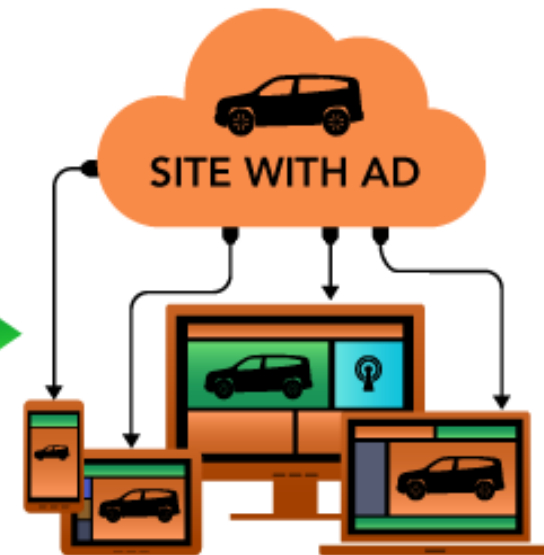
2

Advertisers submit bids to have their ad displayed

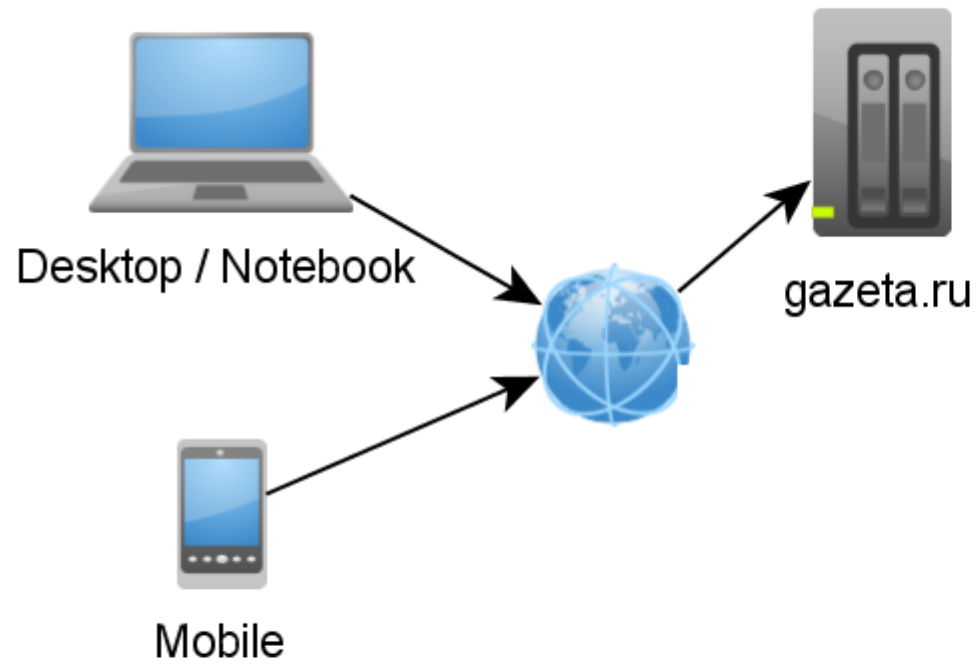


3

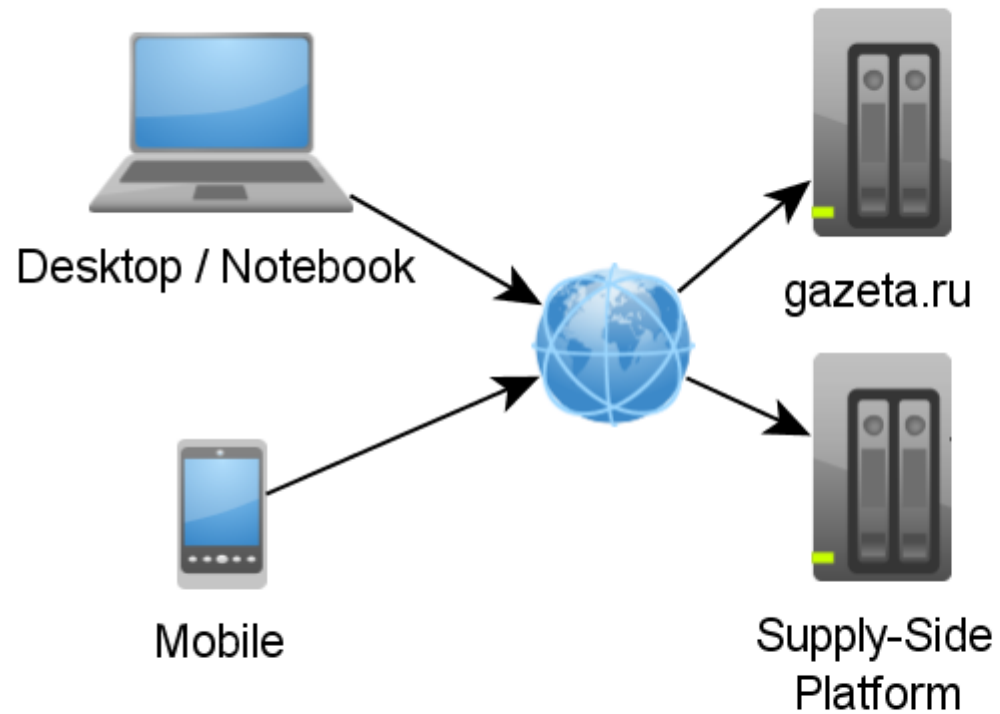
Webpage loads with winner's ad displayed



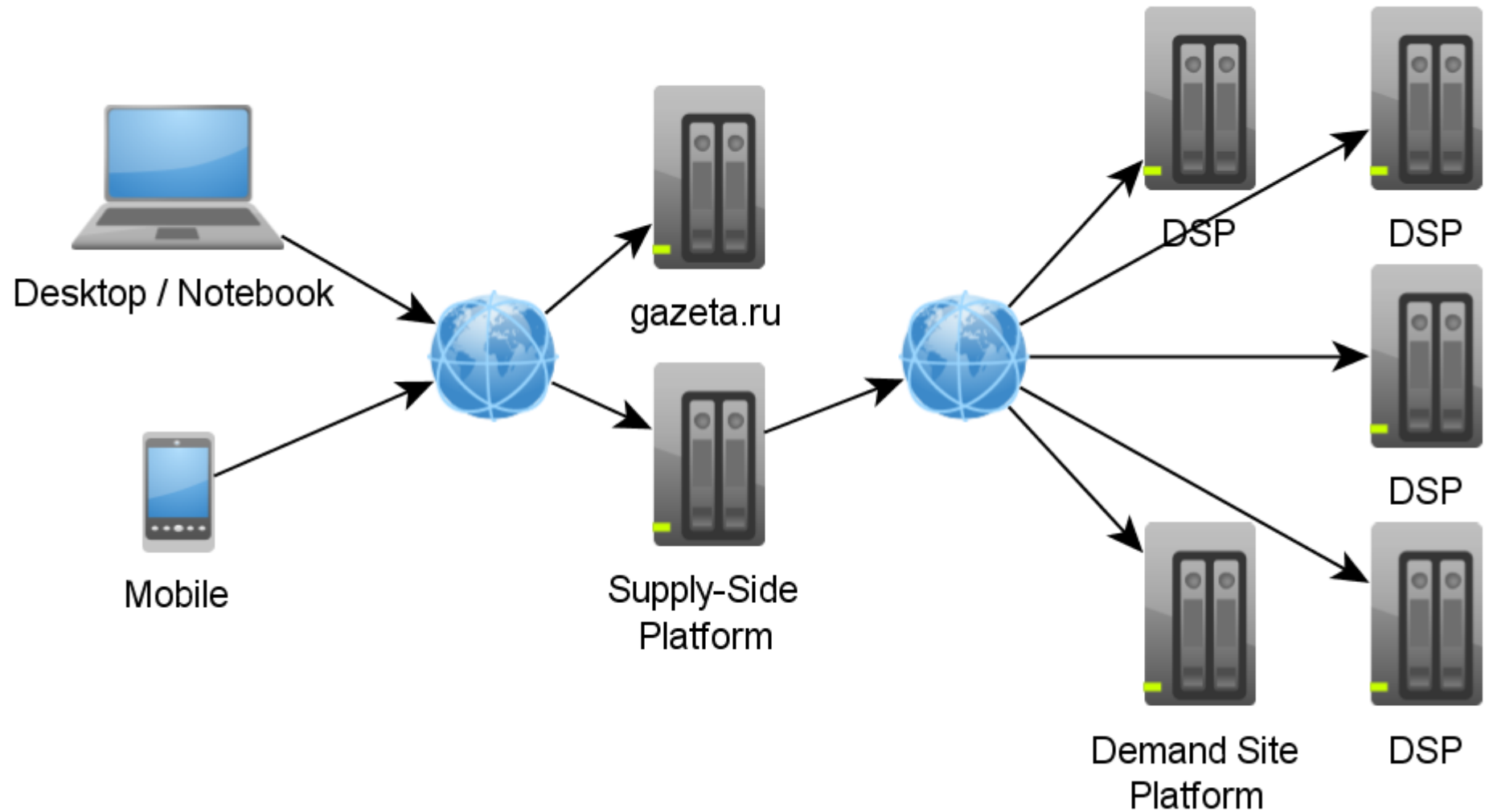
RTB



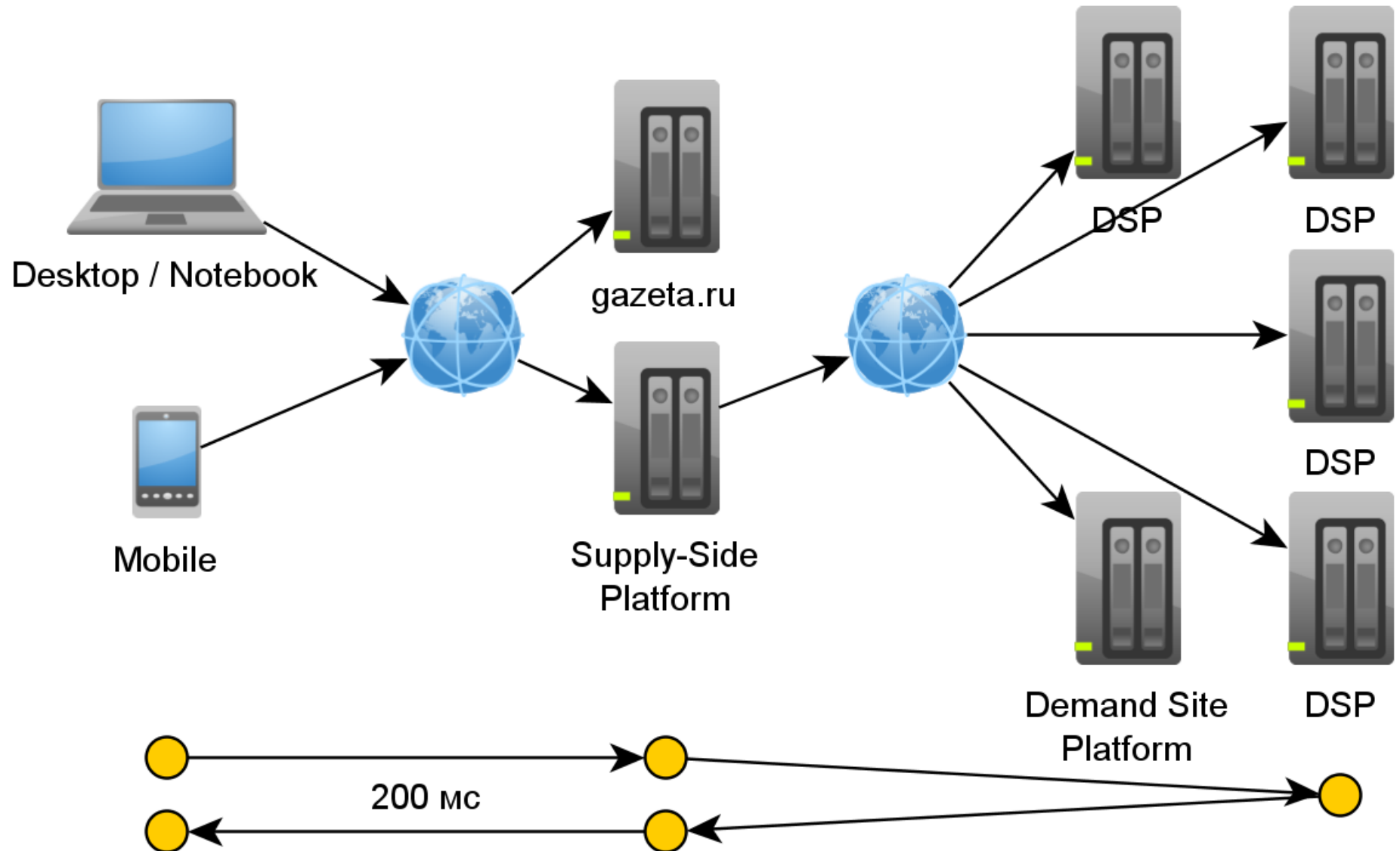
RTB



RTB



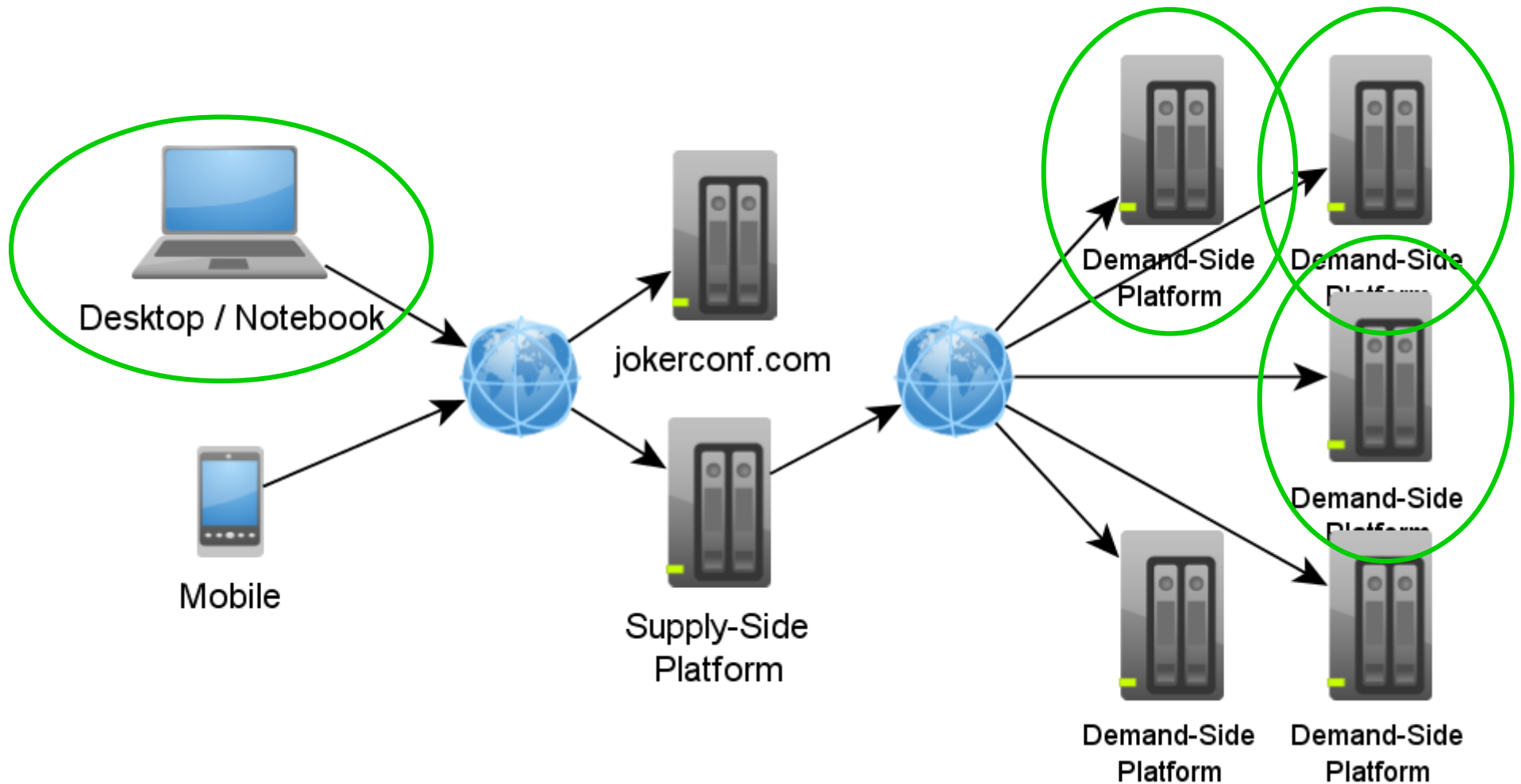
RTB



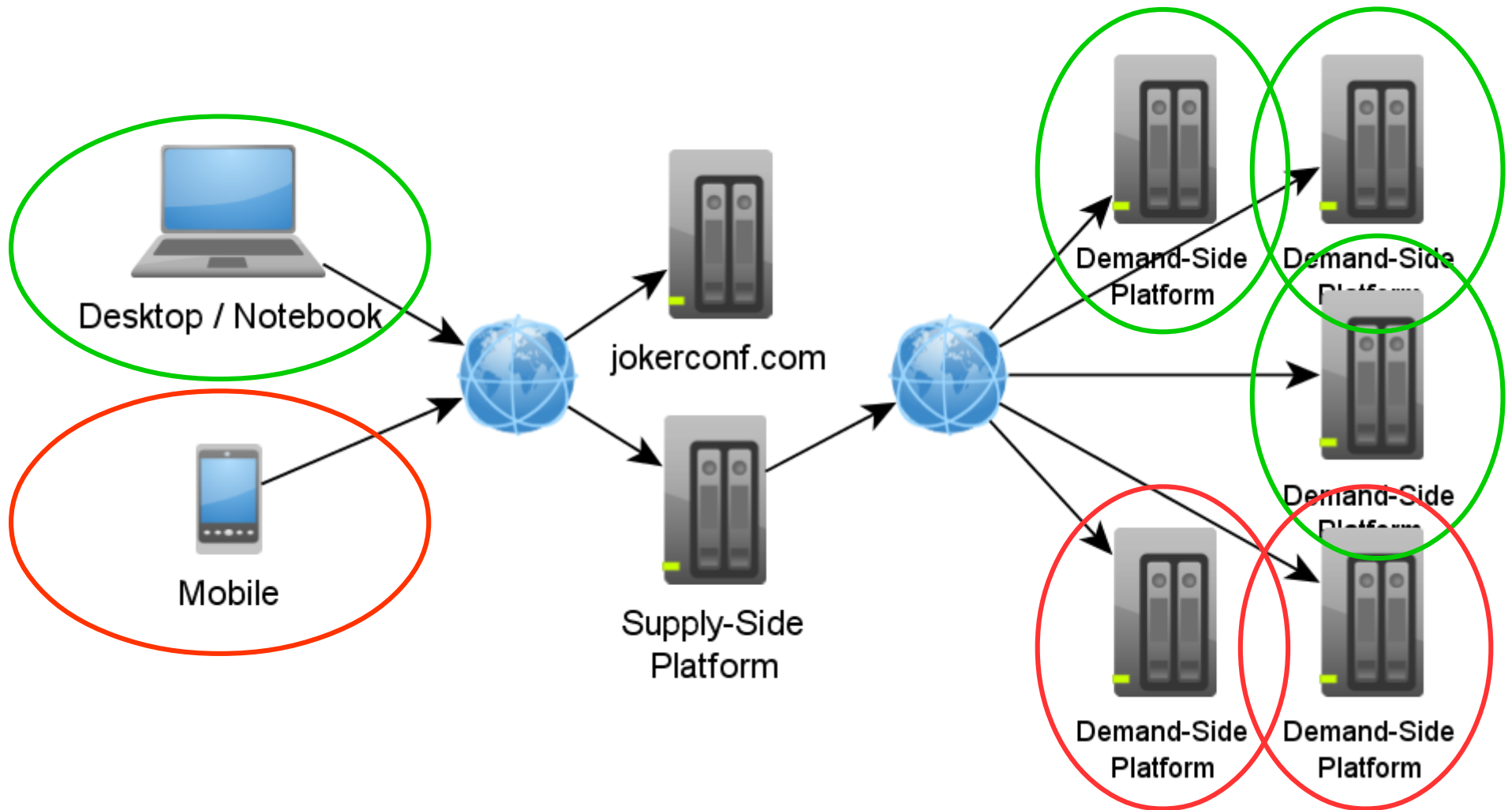
Типичный "SSP"-сервер

- 10 тыс. обращений в секунду
- отношение "входящего" трафика к исходящему от 1 к 5 до 1 к 25
- это когда stop-the-world = потерянные деньги

Предварительная фильтрация



Предварительная фильтрация



Предварительная фильтрация

- Отделить ~~зёрна от плевел~~ настольные браузеры от мобильных

Предварительная фильтрация

- Отделить ~~зёрна от плевел~~ настольные браузеры от мобильных
- Проект «Browser Capabilities Project»
<https://browscap.org/>

Предварительная фильтрация

[360Spider]

Parent=DefaultProperties

Comment=360Spider

Browser=360Spider

Browser_Maker=Qihoo 360 Technology Co. Ltd.

Crawler=true

[Mozilla/4.0 (compatible; MSIE 8.0*; *Windows NT 5.1*Trident/4.0*)* 360Spider]

Parent=360Spider

Platform=WinXP

Device_Type=Desktop

Device_Pointing_Method=mouse

Предварительная фильтрация

- Строка UserAgent:
 - Mozilla/5.0 (Windows NT 6.3; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
- Шаблон:
 - Mozilla/5.0 (*Windows NT 6.3*WOW64*) Gecko* Firefox/37.0*

Предварительная фильтрация

- Проблема... 287261 шаблон

Pattern Matching

```
for (String pattern : db.keySet()) {  
    lines.put(pattern, toPattern(pattern));  
}
```

```
Matcher matcher = PATTERN_EMPTY.matcher(useragent);  
for (final Map.Entry<String, Pattern> entry :  
                                     lines.entrySet()) {  
    matcher.usePattern(entry.getValue());  
    if (matcher.matches()) {  
        return entry.getKey();  
    }  
}  
return "*";
```

Как оценить скорость?

- взять и запустить?

Как оценить скорость?

- Взять и запустить?
 - а как же JIT?

Как оценить скорость?

- Взять и запустить?
 - а как же JIT?
 - а как же GC?

Как оценить скорость?

- Взять и запустить?
 - а как же JIT?
 - а как же GC?
 - а как же (любая другая технология внутри JVM, из-за которой может поменяться скорость выполнения кода)

Используем JMH

- JMH is a Java harness for building, running, and analysing nano/micro/milli/macro benchmarks written in Java and other languages targetting the JVM.
 - <http://openjdk.java.net/projects/code-tools/jmh/>

Используем JMH

```
@State(Scope.Benchmark)
public class Benchmark {

    private BrowsersCapParser1 parserRegexp;
    private String userAgent = "Mozilla/5.0 (Windows NT 6.1;
WOW64; rv:47.0) Gecko/20100101 Firefox/47.0";

    @Setup
    public void setup() throws Exception {
        parserRegexp = new BrowsersCapParser1();
        parserRegexp.init();
    }

    @org.openjdk.jmh.annotations.Benchmark
    public String testRegexp() throws Exception {
        return parserRegexp.getPattern(userAgent);
    }
}
```

Pattern Matching

Result "testRegex":

4,404 $\pm$ (99.9%) 0,047 ops/s [Average]

(min, avg, max) = (3,869, 4,404, 4,951), stdev = 0,198

CI (99.9%): [4,357, 4,451] (assumes normal distribution)

- 4,4 операций в секунду ~ 227 мс на "операцию"

Pattern Matching

- 227 мс на поиск шаблона...
- Кто виноват и что делать?

Другое регулярное выражение?

- Вместо **287261** шаблонов использовать один, но большой:

$(^pattern1\$) | (^pattern2\$) | \dots | (^.*\$)$

- Поиск *группы* в шаблоне:

```
final Matcher matcher = pattern.matcher(userAgent);  
if (matcher.matches())  
    for (int i = 0; i < matcher.groupCount(); i++)  
        if (matcher.group(i) != null)  
            return this.allPatterns.get(i);  
return "*";
```

Другое регулярное выражение?

Result "testBigPattern":

5,939 $\pm$ (99.9%) 0,280 ops/s [Average]

(min, avg, max) = (5,621, 5,939, 6,515), stdev = 0,262

CI (99.9%): [5,659, 6,219] (assumes normal distribution)

- 5,9 операций в секунду ~ 168 мс на "операцию"
- на 35% быстрее!

Другую библиотеку?

- TCL/HSRE RegEx in Java
- <https://github.com/basis-technology-corp/tcl-regex-java>
- "in drop" замена за исключением экранирования
СИМВОЛОВ: «\\» ВМЕСТО «\»

Другую библиотеку?

java.lang.StackOverflowError

```
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)
```

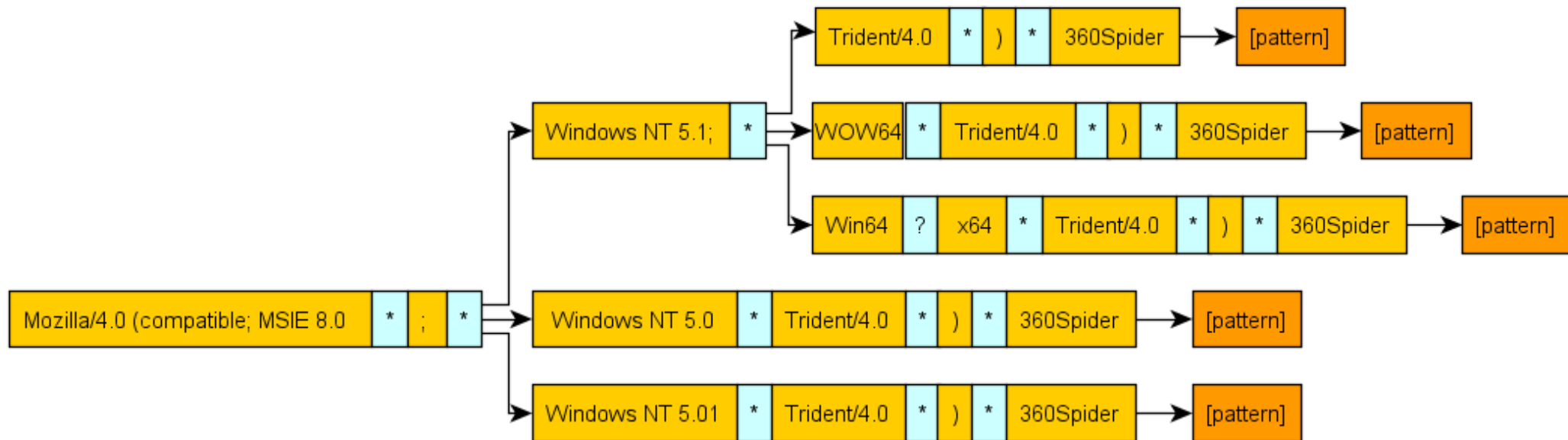
Pattern Matching

- ~~как-то по хитрому использовать regular expression?~~
- ~~взять другую библиотеку?~~
- написать самому?

Pattern Matching

Mozilla/4.0 (compatible; MSIE 8.0	*	;	*	Windows NT 5.01	*	Trident/4.0	*	)	*	360Spider				
Mozilla/4.0 (compatible; MSIE 8.0	*	;	*	Windows NT 5.0	*	Trident/4.0	*	)	*	360Spider				
Mozilla/4.0 (compatible; MSIE 8.0	*	;	*	Windows NT 5.1;	*	Win64	?	x64	*	Trident/4.0	*	)	*	360Spider
Mozilla/4.0 (compatible; MSIE 8.0	*	;	*	Windows NT 5.1;	*	WOW64	*	Trident/4.0	*	)	*	360Spider		
Mozilla/4.0 (compatible; MSIE 8.0	*	;	*	Windows NT 5.1;	*	Trident/4.0	*	)	*	360Spider				

Pattern Matching

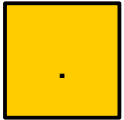


Trie

```
class Trie {  
    final Node root = new Node(' ');  
}
```

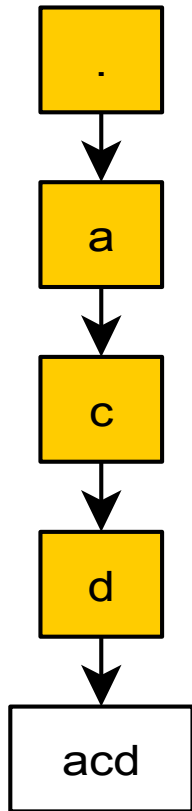
```
class Node {  
  
    final char c;  
    Map<Character, Node> children;  
  
    String leaf;  
}
```

Trie: populate



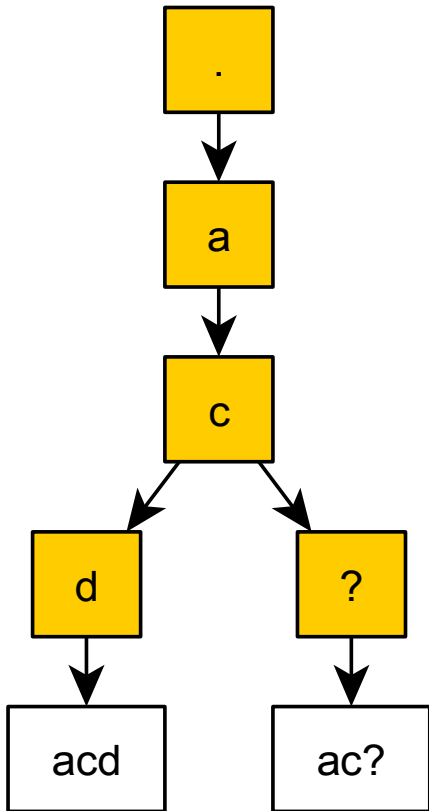
Trie: populate

- "acd"



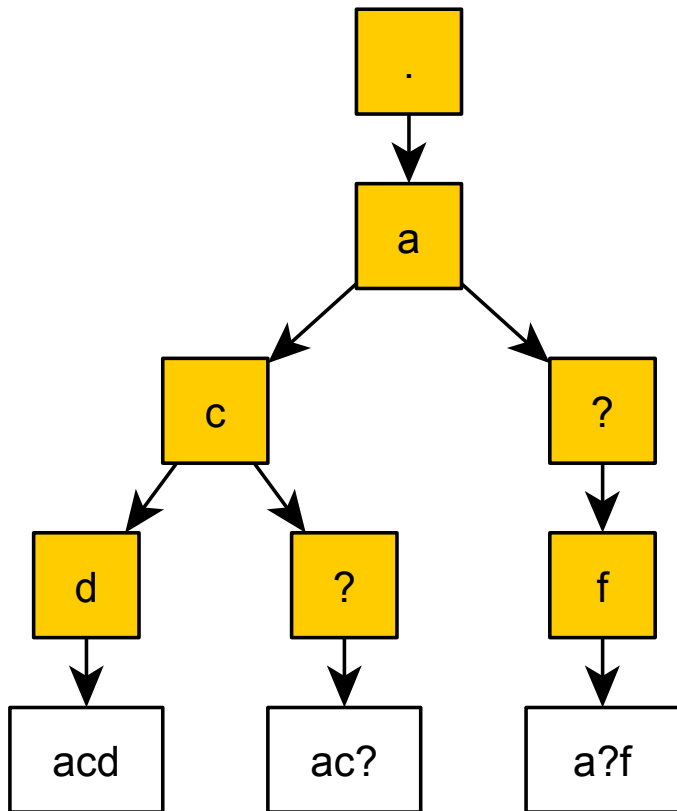
Trie: populate

- "acd"
- "ac?"



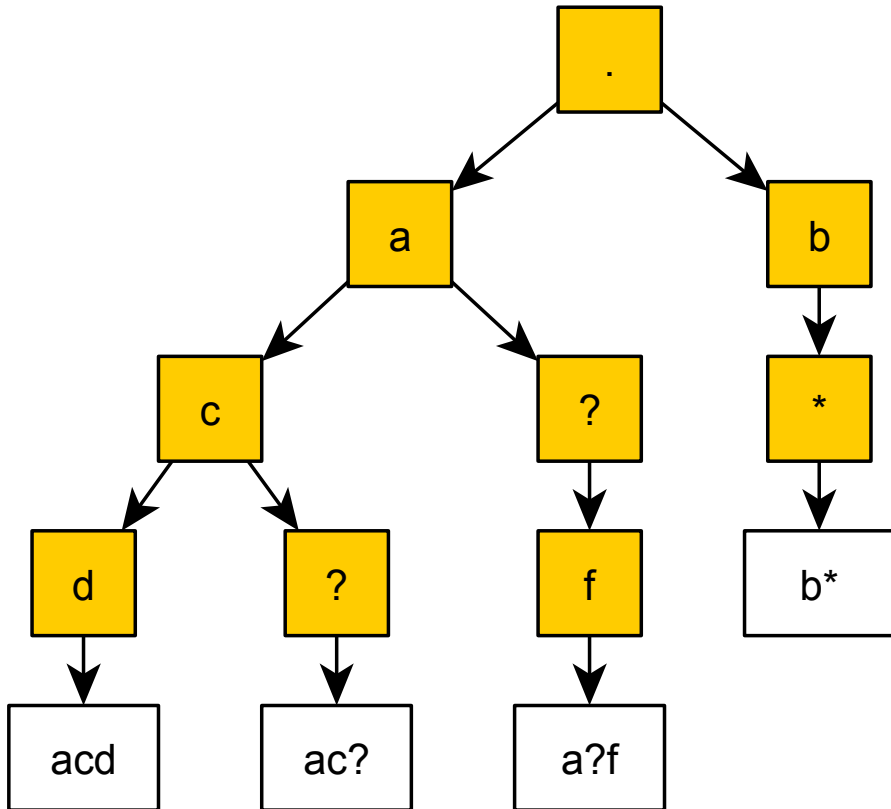
Trie: populate

- "acd"
- "ac?"
- "a?f"



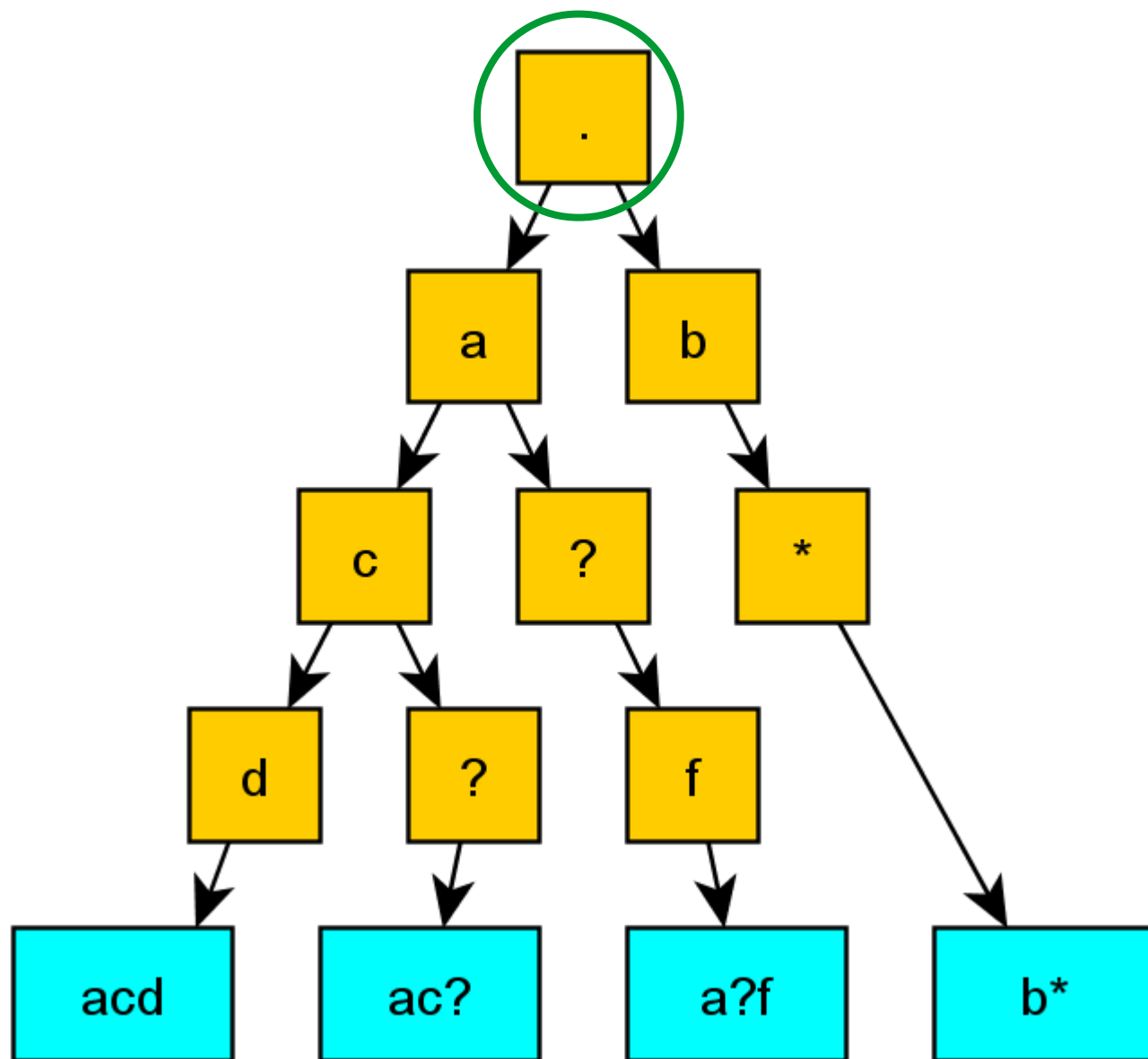
Trie: populate

- "acd"
- "ac?"
- "a?f"
- "b*"

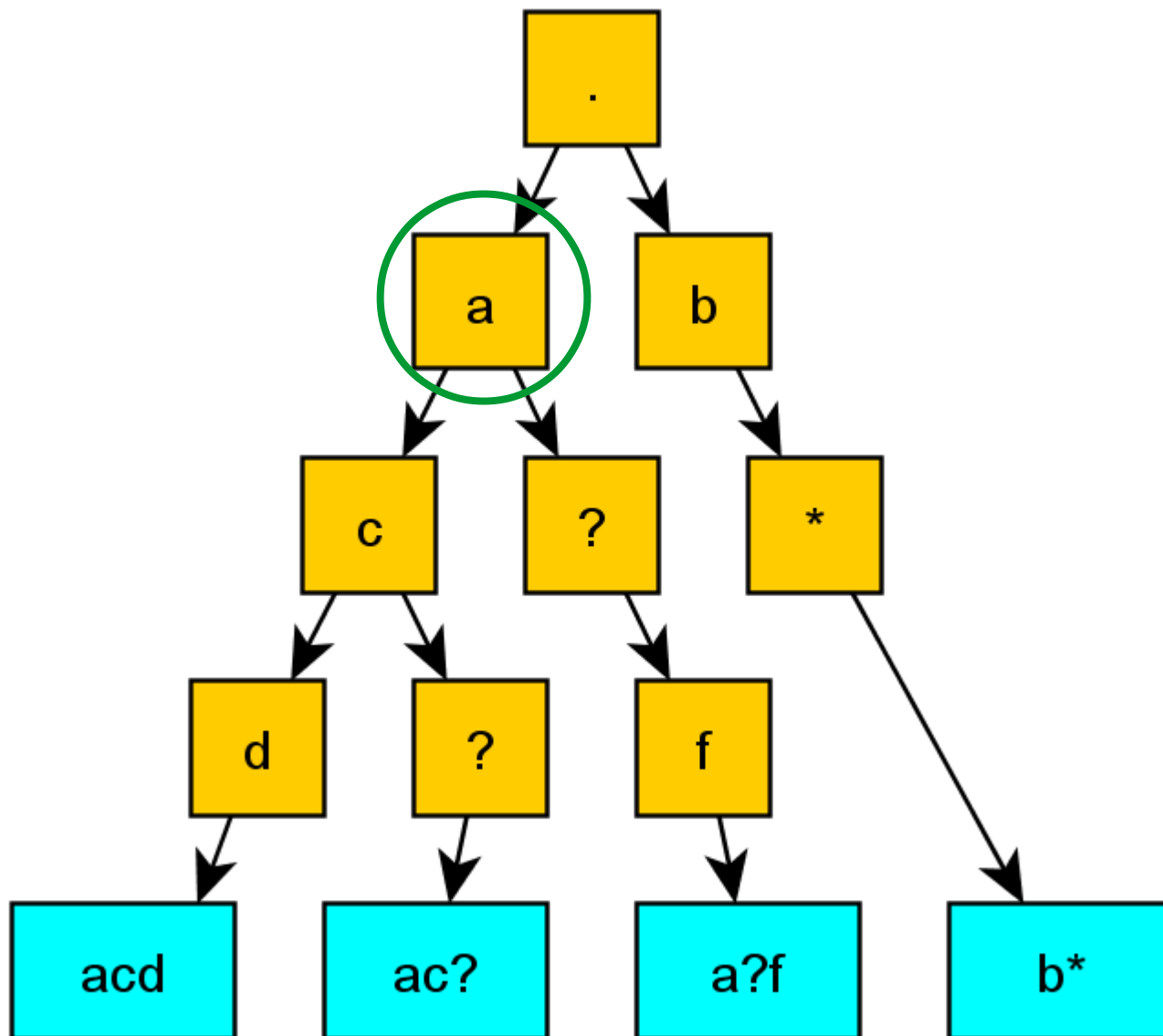


Trie: search

ВХОД: "acf"



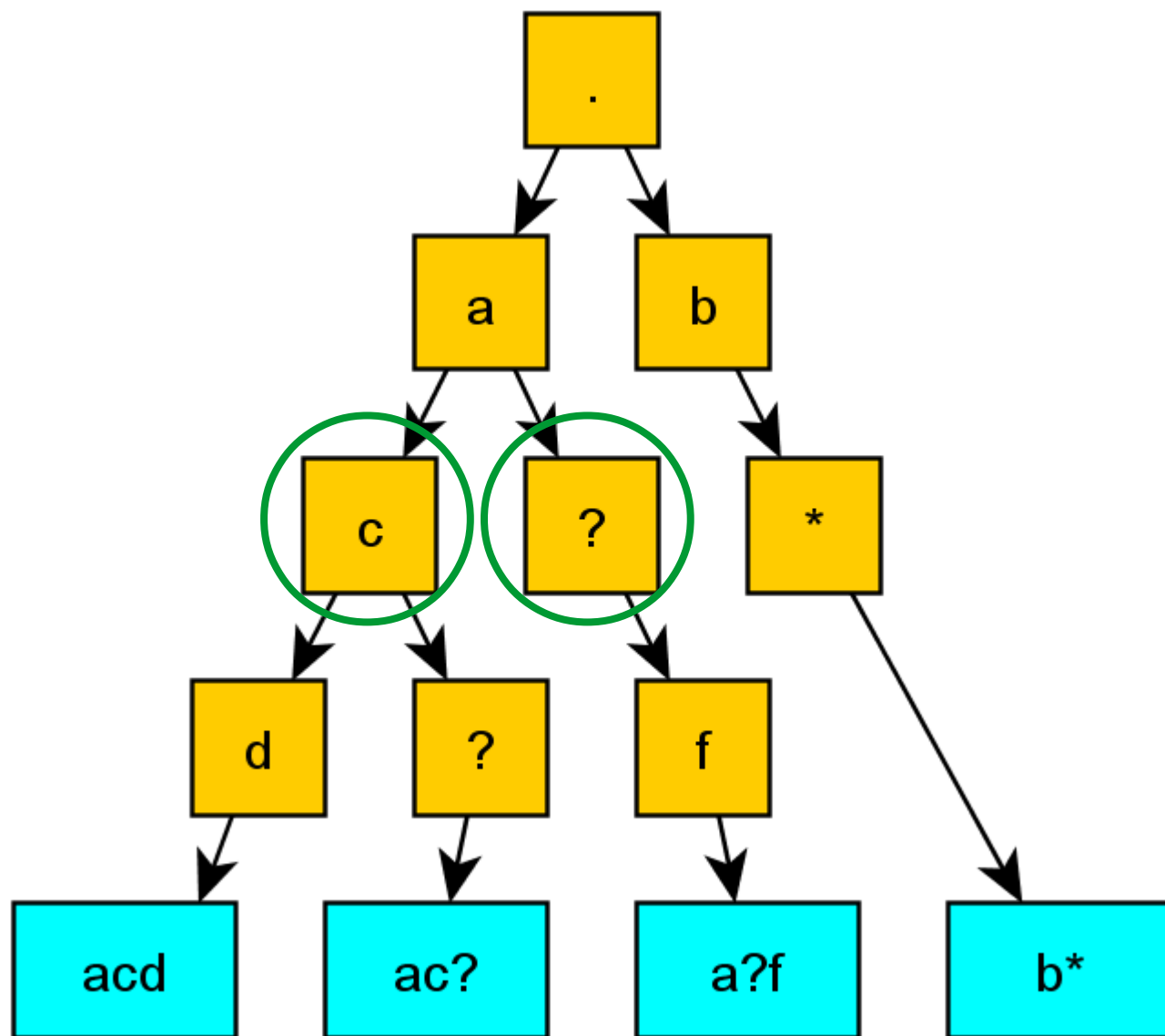
Trie: search



ВХОД: "acf"

1. "a"

Trie: search

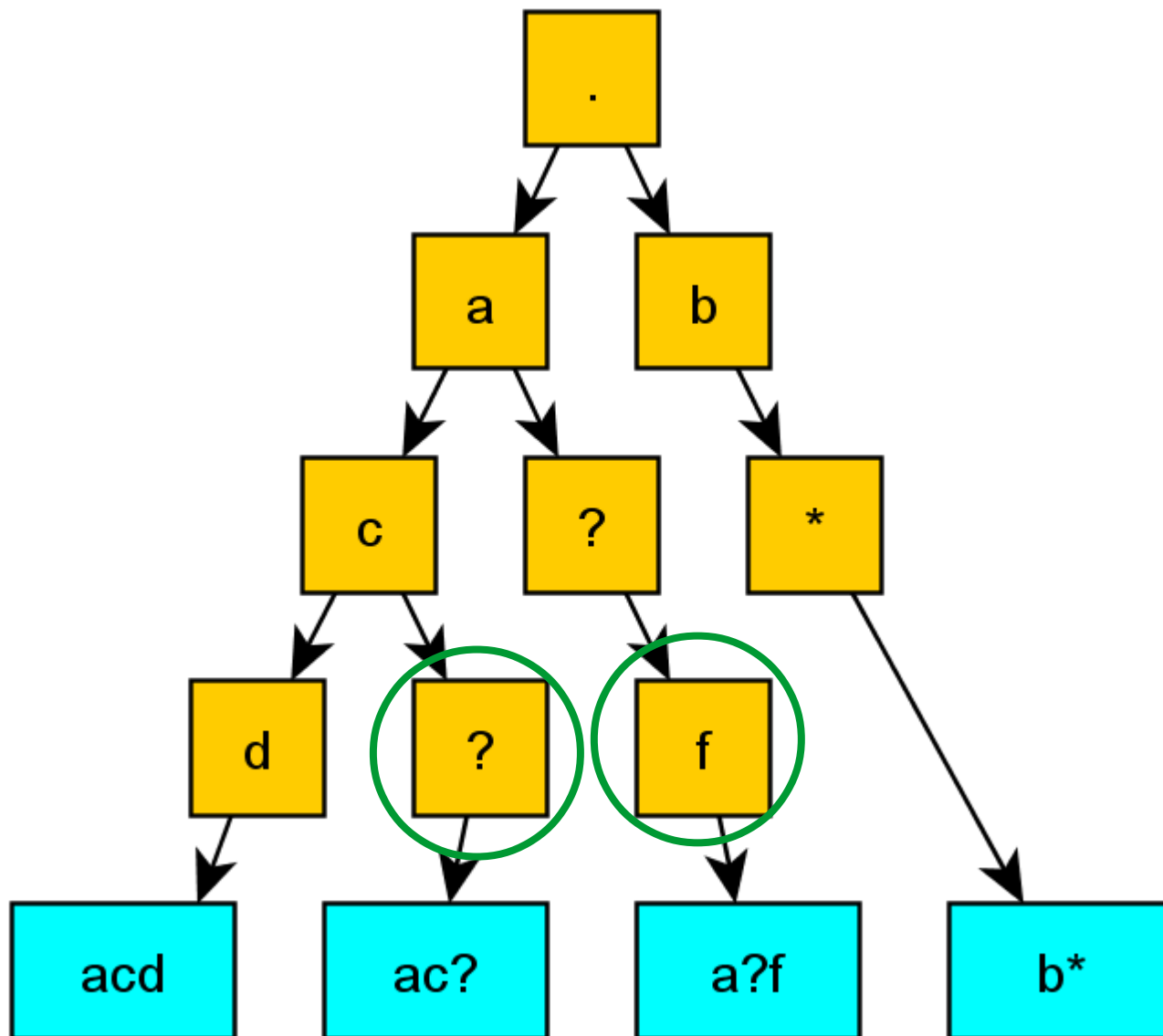


ВХОД: "acf"

1. "a"

2. "c"

Trie: search



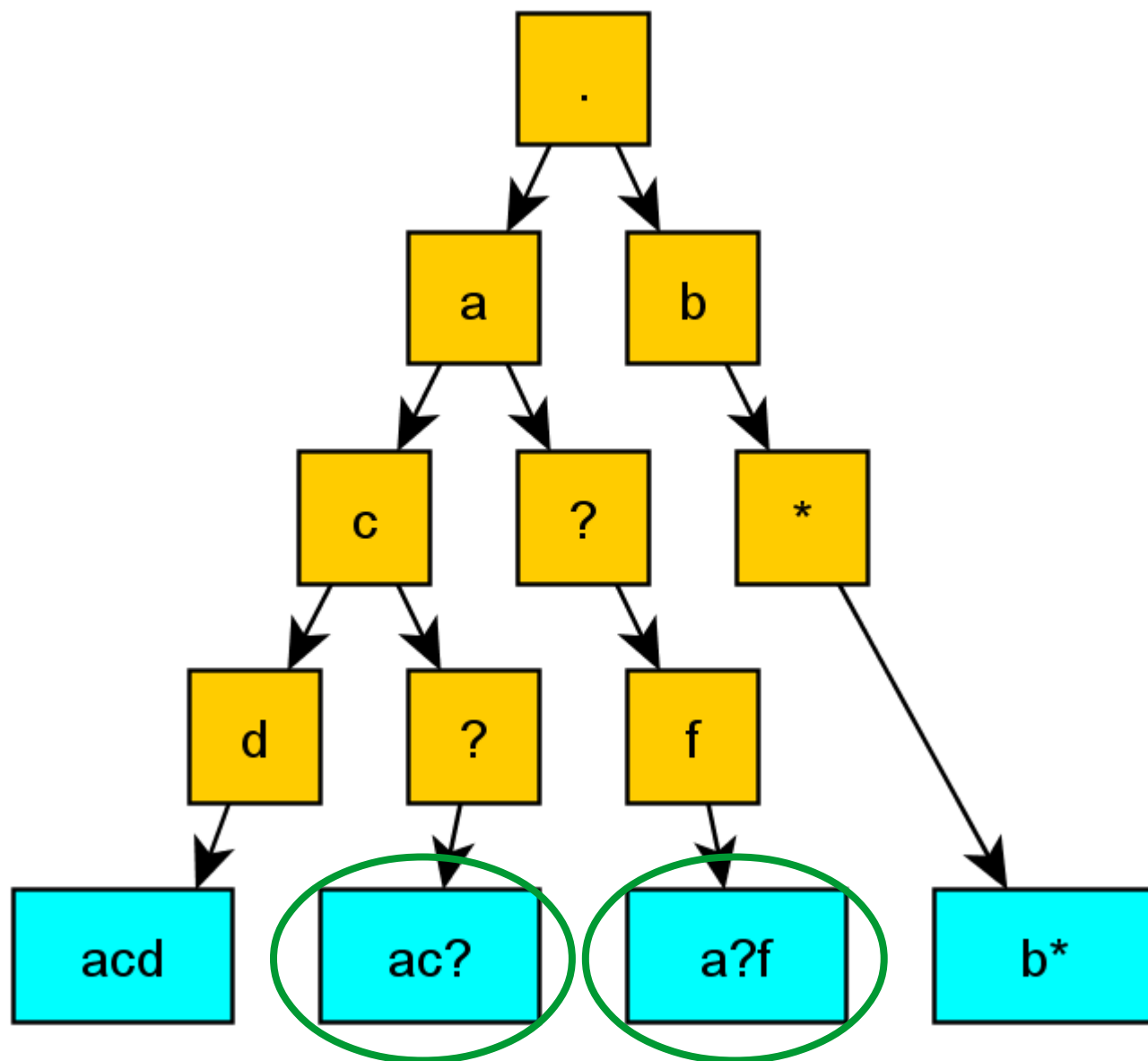
ВХОД: "acf"

1. "a"

2. "c"

3. "f"

Trie: search



ВХОД: "acf"

1. "a"

2. "c"

3. "f"

4. <EoS>

Pattern Matching

- RegEx: $4,404 \pm 0,047$ ops/s
~ 227 мс / строка
- BigPattern: $5,939 \pm 0,280$ ops/s
~ 168 мс / строка
- Trie: $10055,164 \pm 125,745$ ops/s
~ 99 мкс. / строка
- ускорение в 2283 / 1693 раза!

Исходные коды

- <http://github.com/vlsergey/Secretary>
- <http://github.com/vlsergey/jconf2016>

вопросы?